НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

«До захисту допущено»

Завідувач кафедри
_____ Надія БУРАУ
(підпис)
“___”_____20__ р.

Дипломний робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»

спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології»

на тему: «Автоматизована система супроводу навчального процесу»

Виконав (-ла):
студент (-ка) IV курсу, групи ПГ-61
Ляховецький Олег Олегович _____

Керівник:

к.т.н., доцент.

Цибульник Сергій Олексійович _____

Рецензент:
Ст. викл., к.т.н

Божко Костянтин Михайлович _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Пояснювальна записка
до дипломної роботи

на тему: Автоматизована система супроводу навчального процесу

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Повна назва інституту/факультету

Повна назва кафедри

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп’ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані технології та системи навігації і керування»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Н.І. Бурау

«___»_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Ляховецькому Олегу Олеговичу

1. Тема роботи «Автоматизована система супроводу навчального процесу», керівник роботи Цибульник Сергій Олексійович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 08.06.2020р.

3. Вихідні дані до роботи

4. Зміст роботи:

1. Огляд автоматизованих систем супроводу навчального процесу.
2. Огляд інформаційних систем та баз даних.
3. Проектування бази даних.
4. Реалізація бази даних.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Дата видачі завдання 02.04.2020р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
	Огляд автоматизованих систем супроводу навчального процесу	13.04.2020	
	Огляд інформаційних систем та баз даних	20.04.2020	
	Проектування бази даних	04.05.2020	
	Реалізація бази даних	25.05.2020	
	Оформлення пояснювальної записки та графічного матеріалу	08.06.2020	

Студент О.О. Ляховецький

Керівник С.О. Цибульник

Реферат

Пояснювальна записка до дипломної роботи складається з 70 сторінок, містить 16 ілюстрацій, 1 додаток, використано 9 джерел.

Актуальність. Створення інформаційних систем для супроводу навчання в час розвитку інформаційних технологій є досить актуальним, так як постійно розвивається навчальний процес, методики навчання та з'являються нові дисципліни. Все це потребує обліку відповідних розробці нових рішень. Використання сучасних технологій у системі навчання викликає нові задачі перед організацією навчального процесу. Одна з таких задач полягає в необхідності зміни традиційної системи навчання, технологій та розробці нових рішень. Навчальна система повинна стати гнучкою, орієнтованою на студента та динамічно змінюватися.

Мета роботи. Метою роботи є розробка бази даних для автоматизованої системи супроводу навчального процесу.

Перелік ключових слів: інформаційна система, СУБД, база даних, SQL.

Abstract

The explanatory note to the thesis consists of 70 pages, contains 16 illustrations, 1 appendix, used 9 sources.

Topicality. The creation of information systems to support learning during the development of information technology is quite relevant, as the educational process, teaching methods and new disciplines are constantly evolving. All this requires taking into account the development of new solutions. The use of modern technologies in the education system raises new challenges for the organization of the educational process. One such challenge is the need to change the traditional learning system, technologies and develop new solutions. The educational system must become flexible, student-oriented and change dynamically.

The goal of the work. The aim of the work is to develop a database for an automated system for monitoring the educational process.

List of keywords: information system, database, database, SQL.

Зміст

Перелік скорочень, умовних позначень

АССНП — автоматизована система супроводу навчального процесу.

ДН — дистанційне навчання.

ЕОМ — електронно-обчислювальна машина.

СУБД — система управління базами даних.

SQL — Structured query language.

Вступ

Розвиток комп’ютерних технологій привело до масового використання автоматизації в різних сферах діяльності, також і в навчанні. Широке поширення в навчальному процесі отримали автоматизована система супроводу навчального процесу.

Автоматизована система супроводу навчального процесу (АССНП) [8] — це сукупність зв’язаних між собою технічних, програмно-алгоритмічних, інформаційно-методотичних засобів, призначених для автоматизації навчального процесу, пошуку та оброки навчальної інформації і проведення контрою знань

У теперішній час вже існують та продовжують створюватись різноманітні програмні засоби, які забезпечують процес автоматизованого навчання і тестування знань з різних дисциплін. Але, розробка подібних програмних засобів не перестає бути актуальною. Так як постійно розвивається навчальний процес, методики навчання та появи нових дисципліни потребує обліку відповідних їм специфічних особливостей.

До переваг використання автоматизованих систем навчання в порівняні з традиційними методами можна віднести [9]:

- навчання більшого потоку студентів в порівняно стислий термін;
- кращий баланс між масовим та індивідуальним підходом до навчального процесу;
- можливість дистанційного навчання;
- підвищення інтересу у студентів до навчального процесу.

Розділ 1 Автоматизовані системи супроводу навчального процесу

Використання сучасних технологій у системі навчання викликає нові задачі перед організацією навчального процесу. Одна з таких задач полягає в необхідності зміни традиційної системи навчання, технологій та розробки нових рішень. Навчальна система повинна стати гнучкою, орієнтованою на студента та динамічно змінюватися.

Сьогоднішня методика викладання передбачає, що викладач щорічно веде одні й ті ж лекції, малюючи одні й ті ж схеми, графіки і т.і., що забирає багато часу. Впровадження нових інформаційних технологій в навчання за допомогою лекцій в електронному виді, презентації, інтернету, автоматизованих систем навчання збільшує ефективність та швидкість процесу навчання, так як такі методи викладання краще засвоюється студентами [7].

Розглянемо одну із нових технологій. Автоматизована система супроводу навчального процесу використовується для автоматизації навчального процесу підготовки студентів, як з викладачем, так і без нього. За допомогою АССНП забезпечується навчання, керування навчальним процесом, підготовка навчальних курсів, тестування знань та результати навчання [8].

Автоматизована система супроводу навчального процесу орієнтована на навчання, в основному направлені на донесення теоретичної частини дисципліни, використовуючи теоретичний матеріал в електронному вигляді, презентаціях та відеоматеріалі[7].

Одним із важливих етапів будь-якого процесу навчання є етап перевірки знань, так як результат на виході даного етапу говорить про готовність студента до подальшого навчання того чи іншого матеріалу.

Однією із найпростіших форм автоматизованої системи оцінювання є форма тестів, де використання комп'ютерних технологій дозволяє викладачу

творчо підходити до створення тестів, а рутинну перевірку відповідей студентів залишити на АССНП [7].

Реалізацією АССНП може бути дистанційне навчання. Дистанційне навчання (ДН) [9]- це навчання на відстані, коли викладач і студент розділені просторово і коли всі або більша частина навчальних процедур здійснюється з використанням сучасних інформаційних і телекомунікаційних технологій.

Дистанційне навчання через інтернет – це навчання, при якому значна кількість навчального матеріалу та взаємодія між викладачем та студентом відбувається за допомогою технічних, програмних засобів з використанням глобальної мережі.

Інтерес у студентів до такого навчання зростає. За даними IDC [9], у США більше ніж 50% вищих навчальних закладів використовують дистанційні онлайн програми, як частину навчального процесу [9].

Відмінною особливістю ДН є надання студентам можливості самим отримувати знання, використовуючи різні інформаційні ресурси такі як [7]: різні бази даних, бази знань, комп'ютерні навчальні та контролюючі системи, відео- та аудіозаписи, презентації та електронні бібліотеки. Також використовуючи, традиційні підручники та методичні розробки, створюється унікальна розподілена середа навчання, доступна кожному [9].

Проведення відеолекцій, відеоконференцій, можливість проведення частих консультацій з викладачем за допомогою телекомунікаційних систем, робить взаємодію між студентом та викладачем більш цікавою ніж традиційна форма навчання.

Також такий вид навчання дозволяє за допомогою телекомунікаційних систем спілкування між студентами, що дозволяє проводити семінари та ділові ігри [9].

В останній час збільшилась популярність використання інтернет технологій в якості технології для дистанційного навчання, що пов'язано зі зростанням можливостей технічних засобів зв’язку та розповсюдженням комп’ютерної мережі інтернет. До переваг використання цих технологій можна віднести [9]:

- Можливість передача файлів будь-якого формату та розміру.
- Можливість швидко змінювати інформацію зі свого робочого місця.
- Зберігання навчальної інформації в пам’яті комп’ютера на будь-який час, та редагування, обробка, друкування цієї інформації.
- Оперативний зворотній зв’язок.
- Можливість доступу до різних джерел інформації.
- Можливість реалізації різноманітних електронних конференцій, у тому числі в режимі прямого включення, аудіоконференцій та відеоконференцій.
- Можливість спілкування з будь-яким користувачем, підключеним до мережі інтернет.
- Можливість перенести інформацію в оффлайн та обробка її в зручний для користувача час.

Інтернет мінімізував часові, просторові та фінансові труднощі в розповсюдженні інформації.

Майже всі навчальні заклади використовую системи автоматизації для ведення документообігу. Для автоматизації цієї проблеми з успіхом використовується [] офісні пакети програм Microsoft Office та подібні. Для супроводу навчального процесу багато навчальних закладів використовують спеціалізоване програмне забезпечення.

1.1 Огляд автоматизованих систем супроводу навчального процесу

Одним із представників такого програмного забезпечення є Google Classroom [1] — це безкоштовний сервіс від Google створений для навчальних закладів з метою спрощення, поширення і класифікації завдань безпаперовим шляхом. Його основна мета - прискорити процес поширення файлів між викладачем та студентом [1].

Основні особливості Google Classroom [2]:

1. Налаштування класу. Для кожного класу створюється свій код, який учні можуть використовувати для приєднання до спільноти. Цей процес усуває необхідність створення попередніх реєстрів (але для користування Classroom все ж необхідно мати Google аккаунт).
2. Взаємодія з Google Drive. При користуванні сервісом автоматично створюється тека “Клас” на диску Google у викладача з новими вкладеннями для кожного створеного класу .
3. Автоматизація. При створенні завдання в форматі Google- документа, сервіс розповсюдить копії всім користувачам які приєднані до даного класу.
4. Терміни. При створенні завдання викладач вказує термін виконання роботи. Коли учень надає завдання до початку терміну, на його документі з'являється статус «Перегляд», що дозволяє вчителям робити сортування.
5. Зворотній зв’язок. Коли студент виконує своє завдання, викладач може забезпечити зворотній зв’язок у той момент, коли студент знаходиться в статусі “Перегляду”. Коли завдання повертається до студента, то студент переключається в статус “Редагування” і продовжує роботу.

6. Зручний огляд. І викладачі, і студенти на головному екрані Google Classroom можуть бачити всі завдання. Це дозволяє контролювати роботу в декількох класах одночасно.
7. Зв’язок. Завдяки поєднанню класних оголошень, створених викладачем, і інтегрованим можливостями коментування завдань, у користувачів завжди є можливість підтримувати зв'язок і бути в курсі статусу кожного завдання [2].
8. Традиційні функції. Добре реалізована можливість публікувати теоретичний матеріал, завдання, виставляти оцінки в журнал та вести облік.

До мінусів даної системи можна віднести відсутність функціоналу для проведення тестування знань, тому користувачі створюють тести в іншому додатку від Google, а саме Google Forms. Він створений для проведення опитувань, але доклавши невелике зусилля опитування перетворюються в тести. [3]

Ще одним гравцем у цій сфері є Moodle. Moodle (Modular Object-Oriented Dynamic Learning Environment, вимовляється «Мудл») [4]- це модульне об'єктно-орієнтоване динамічне навчальне середовище, яке називають також системою управління навчанням, системою управління курсами, віртуальним навчальним середовищем або просто платформою для навчання, яка надає викладачам, учням та адміністраторам дуже розвинутий набір інструментів для комп'ютеризованого навчання, у тому числі дистанційного. Moodle можна використовувати в навчанні школярів, студентів, при підвищенні кваліфікації, бізнес-навчанні, як у комп'ютерних класах навчального закладу, так і для самостійної роботи вдома [4].

Moodle - це програмне забезпечення з відкритим вихідним кодом, тобто таке програмне забезпечення можливо безкоштовно завантажувати,

використовувати, та змінювати або розширювати. До плюсів можна віднести повністю безкоштовне використання, створення якісних курсів, можливість використання навчальної інформації в різних форматах - аудіо, відео, текст та інших.

Також в Moodle можливо підключати різні додатки від інших користувачів наприклад [3]:

- Модуль відеоконференції.
- Масові розсилання сповіщень.
- Аудіо та відео чати та інші.

До мінусів можна віднести [3]:

- Для використання системи потрібен пристрій для розгортання (сервер або хостинг).
- Moodle має досить високі системні вимоги.
- Потребує вивчення - метод “научного тика” не пройде.

Завдяки плагінам та веб-розробці кожна система на moodl може виглядати унікально, налаштована під конкретні цілі.

Наступна АССНП - це web-додаток Edmodo [3], тому що сервіс не потрібно ніде встановлювати. Edmodo позиціонує себе як соціальна мережа для навчання або Facebook для навчання - він побудований за принципом соціальних освітніх мереж, та й інтерфейс нагадує зовнішній вигляд Facebook.

Логіка роботи в даному додатку наступна: учитель створює групу. Група має своє унікальну посилання і код, які потрібно повідомити іншим учасникам освітнього процесу. Група може мати такі навчальні елементи, як: записи (у вигляді тесту або файлів), тести, завдання і опитування. Можна імпортувати контент з інших сервісів, наприклад новинні стрічки із сайту навчального закладу, відео з YouTube, контент з інших сервісів.

Особливих можливостей в Edmodo немає, але є прості і потрібні елементи - календар (для фіксації навчальних подій, журнал для виставлення оцінок, можливість перевірки домашнього завдання і т.і.).

Виділимо переваги сервісу:

- безкоштовний;
- немає реклами;
- проста реєстрація;
- користувачі діляться на три групи: вчителі, учні, батьки (у кожної групи своя окрема реєстрація, свій код для доступу).

До недоліків відносяться:

- Відсутня українська мова — хоча інтерфейс простий та зрозумілий.
- Неможливо об’єднати групи.
- В цілому арсенал навчальних елементів хоч і достатній, але відносно бідний - ті ж тести не містять додаткових стратегій, немає тематичних тестів і т.і..

iSpring Learn [3] — це інтернет-сервіс, а значить не потрібно завантажувати програму, встановлювати на сервер, налаштовувати. Щоб почати роботу, досить зареєструватися на сайті, завантажити навчальні матеріали та призначити співробітникам.

Особливості iSpring Learn [3]:

- Безлімітне сховище. У системі можна завантажити необмежену кількість навчальних матеріалів: курсів, відеороликів, книг, презентацій.
- Редактор курсів в PowerPoint. У iSpring є редактор, в якому можна зробити електронний курс з PowerPoint-презентації з відео, тестами, інтерактивними іграми.

- Мобільне навчання. Курси можна відкрити на комп'ютері, планшеті, смартфоні навіть офлайн, наприклад, в поїзді або літаку. Є мобільні додатки для android і iOs.
- Детальна статистика. Система збирає детальну статистику і допомагає контролювати успішність співробітників. Звіти показують, які курси користувач завершив, який прохідний бал набрав, скільки допустив помилок в тесті. Всього 11 типів звітів, які можна завантажити в .xls та .csv для подальшої обробки.
- Вебінари. У платформу інтегрований професійний сервіс для відеоконференцій і вебінарів Zoom. Можна демонструвати робочий стіл, презентацію або відео, писати в загальний і особистий чат. Система автоматично відправляє учасникам нагадування про найближчу онлайн-зустріч і повідомляє про зміну в розкладі - писати кожному особисто не доведеться. Записи вебінарів зберігаються.
- Установка на сервер клієнта. Провайдер готовий установити платформу на сервер замовника, у цьому випадку оплата за платформу буде одноразова. Фахівці з iSpring будуть оновлювати платформу до актуальної версії раз на рік.

До недоліків можна віднести:

У iSpring Learn є безкоштовна 14-денна пробна версія, але в цілому система платна. Однак на безкоштовну систему автоматизації навряд чи піде менше грошей: доведеться витратитися на свою підтримку, найняти програмістів для її адміністрування.

Оренда платформи. Вартість платформи залежить від кількості активних користувачів на платформі. Чим більше тариф - тим менше ціна за користувача.

У випадку з платною платформою надається повний сервіс: допоможуть розвернути і налаштувати навчальної портал, завантажити матеріали і почати

навчання співробітників. Будь-яке питання співробітники техпідтримки вирішать по телефону.

Системи дистанційного навчання з відкритим вихідним кодом дають можливість створювати і ефективно розвивати курси електронного навчання, особливо якщо ви готові витратити деякий час на доскональне вивчення всіх можливих функцій системи. У деяких випадках, використання подібних систем може позначитися на кривій навченості, але економія коштів і свобода у виборі зовнішнього вигляду і наповненості курсу, в кінці кінців, покриває всі можливі труднощі.

Крім готових систем використовуються і технології які не завязані на конкретному додатку. Одною з таких технологій є QR-код. QR (quick response) [6] у перекладі з англійської “Швидка відповідь”. Це двомірний штрих-код (матричний код), який розробила японська компанія "Denso Wave" в 1994 році. Він дозволяє в одному невеликому квадраті помістити 2953 байта інформації, тобто 7089 цифр або 4296 букв (близько 1-2 сторінок тексту в форматі А4).

QR-коди вже активно використовуються музеями і видавництвами для кодування додаткової інформації про об'єкти культурної та історичної спадщини і розміщення активних посилань на свої сайти (з можливістю переходу по ним), туристичними компаніями для розміщення на туристичних об'єктах інформації на різних мовах, компаніями-виробниками для розміщення як інформації про товари, так і своїх даних.

У навчанні QR-код можна використати наступним чином:

- Кодувати завдання для групової чи індивідуальної роботи;
- З легкістю поширювати посилання на різні інформаційні, мультимедійні ресурси, які зберігають додаткову інформацію;
- Контролювати відвідування лекцій.

Так, наприклад, використання даної технології використовується в китайських університетах [6]. Під час заняття на дошці з'являється QR-код, який студенти повинні зчитати через WeChat, для підтвердження своєї присутності. WeChat— мобільна платформа для обміну текстовими та голосовими повідомленнями, розроблена компанією Tencent [5]. Система використовується, але не досить ефективно, таку систему можливо обійти зчитавши QR-код з фото, яке отримане від інших учасників навчального процесу [6].

Використання окремо технологій має великий недолік: відсутність єдиного підходу, що зменшують ефективність та переваги кожної з цих розробок. Негативно впливає і той факт, що програми від різних розробників не можуть здійснювати ефективний обмін даними між собою.

Тому, більшість навчальних закладів схильні до використання інтегрованих систем управління навчальним процесом, які дозволяє автоматизувати всі необхідні задачі.

1.2 Постановка задачі

При правильному підході до розробки будь-яких систем безпосередньо перед моделюванням потрібно виділити основні вимоги до проекту. У випадку з розробкою автоматизованої системи супроводу навчального процесу необхідно зрозуміти, які завдання ставляться перед системою, яким чином інтегрувати її в процес навчання і які ресурси і можливі витрати на проектування і підтримку. Але перш за все, важливо проаналізувати стан освітнього процесу, на базі якого будується система, а саме: виділити вихідні дані, необхідні для розробки. На даний момент у навчальному процесі існують процедури, які конкретно не впливають на навчання, але необхідні для його забезпечення. Так, наприклад, проведення тесту. Тест це важлива частина навчання, але проведення та перевірка тесту займає багато часу, тому що більшість викладачів проводять його “вручну”. Що мається на увазі “вручну”? Тобто, готують завдання потім роздають у надрукованому вигляді студентам, що займає час, або диктують, що

займає більше часу. Потім студенти проходять тест на листках і повертають листи з відповідями викладачу, після чого викладач перевіряє роботу кожного студента. Цю процедуру можна автоматизувати за допомогою інформаційних систем. Так, наприклад, викладач може створити тест в програмному забезпеченні, яке встановлено на його комп'ютері і прийшовши на пару одночасно роздати тест студентам зі свого смартфону. Студенти в свою чергу складають тест і моментально дізнаються свою оцінку, яку також бачить викладач. У такому сценарії зберігається час при роздачі та перевірці тесту. Цей час викладач може витратити на більш детальну розробку свого предмету.

Також невід'ємною частиною навчального процесу є присутність студента на парі та перевірка його присутності. Зараз контроль присутності проводиться в більшості випадків за допомогою "переклички" або за допомогою старости, який повідомляє викладача про присутність у групі. "Перекличка" займає досить багато часу, особливо на потокових лекціях, де присутні 40 і більше студентів, а інформація від старости не завжди вірна. Автоматизувати даний процес складніше. Так наприклад існує спосіб автоматизації за допомогою QR-коду, але мінус такого варіанту в тому що цей QR-код потрібно десь відобразити та відвести час на сканування. Але головним мінусом є те що студент може зафотографувати цей QR-код та відправити одногрупникам, які не знаходяться на парі. Тому потрібні більш захищені технології, або технології, які не мають прямої взаємодії з користувачем. Один із варіантів це WI-FI. Викладач приносить пристрій, який розповсюджує точку доступу і студенти підключаються до неї. Усі користувачі, які підключені до пристрою вважаються присутніми. Плюси такого варіанта в тому, що для підтвердження присутності потрібно перебувати в зоні дії точки доступу. А мінусом є те, що потрібно додаткове апаратне забезпечення. Ще одним варіантом є визначення присутності за місце знаходженням. У такому випадку дані про місце знаходження студента, на період пари, передаються на сервер, де порівнюються з місцем знаходження викладача і якщо вони однакові з певним

допуском то вважається, що студент присутній. Перевірка може відбуватися декілька разів за пару або викладач може встановлювати коли та скільки разів перевіряти присутність.

Отже система повинна мати:

- можливість проводити тести;
- автоматизувати процес контролю присутності.

Розділ 2 Моделювання автоматизованої системи супроводу навчального процесу

2.1 Інформаційні системи та бази даних

Бази даних беруть свій початок з інформаційних систем, які в наш час оточують нас всюди. Ми зустрічаємо інформаційні системи коли використовуємо соціальні мережі, телефонуємо з мобільного телефона, бронюємо квитки або номера в готелях, на касі супермаркету розплачуючись кредитною карткою або PayPass, роблячи покупки в інтернеті, користуючись банкоматом, водячи пошуковий запит в Google таких прикладів можна приводити до нескінченності.

Що таке інформаційна система? Це складна багато функціональний програмно-апаратний комплекс, забезпечуючий наступні функції. В першу чергу, це надійне збереження інформації. Друга функція це виконання специфічних для даного додатка обробка інформації на обчислень, реалізуючих функції бізнес логіки. Третьою функцією кожна інформаційна система дає користувачу зрозумілий та зручний інтерфейс, в якому можуть вводити інформацію та отримувати зрозумілий результат.

Історія інформаційних систем включає в себе лише десятки років на відміну від фундаментальних наук які включають багатовікову історію. В 60-ті роки інформація яка оброблювалась в інформаційних системах стала використовуватись в звітності по багатьом параметрам. В ті роки головним замовником інформаційних систем були банки, які розробляли банківські системи. В 70-ті підприємства починають використовувати інформаційні системи для управління виробництвом, які могли підтримати та прискорити виробництво.

Інформаційні системи тих років здебільшого створювались для вирішення конкретних задач, які чітко визначалися на початковому етапі та не мінялись в

життєвому циклі інформаційної системи. В той період почали з'являтися персональні комп'ютери, доступні не лише великим компаніям чи університетам, а й звичайним користувачам.

Для персональних комп’ютерів починає розроблятись незлічена кількість різноманітних інформаційних систем, і бази даних створюються для вирішення різних взаємопов'язаних задач в які дані надходять з різних джерел.

В ті часи розвивалися багато мов програмування серед них також розвивалися мови для баз даних. Популярними архітектурами тогочасних баз даних були дві моделі. Перша - це модель локальної інформаційної системи, яка базувалась на персональному комп'ютері якою користувався один користувач. Друга модель - це модель якою могли використовувати велика кількість користувачів, такі бази даних розгорталися на великих ЕОМ в ті часи їх називали ЕОМ третього та четвертого покоління. При використання такої моделі користувачі підключались до системи через термінал.

До швидкого розвитку баз даних призвело розуміння, що дані важливіші ніж програма на якій вони обробляються. Раніше було так: на вхід будь-якої програми подавалася інформація, оброблювалася, видавався результат на цьому закінчився життєвий цикл програми та даних які вона оброблювала. Але з'явилася потреба в збережені, захисті, доступності для багатьох користувачів до таких даних. А програми можуть змінюватись, видалятись, створюватись нові. Ще однією причиною розвитку баз даних став розвиток потужних обчислювальних систем. Так як для обробки великої кількості даних потрібна велика обчислювальна потужність. Так за законом Гордона Мура, один із засновників компанії Intel, який стверджує, що обчислювальна потужність комп’ютерів збільшується в двоє кожні два роки. Також на розвиток вплинули зростання об'єму носіїв інформації та зміна їх типів. До 1960 року носії були послідовні. Це були барабани чи стрічки доступ до інформації, яка зберігається на них, можна було отримати перемотавши до конкретного місця. Пізніше

з’явились пристрої прямого доступу до даних, що дозволило здійснювати вибіркове оновлення складних структур даних.

У розвитку баз даних велику роль отримала поява мережі інтернет. Яка з'явилася в 1969 розі і мала назву ARPANET. Пізніше це перетворилось в гігантську інформаційну систему, в які не переставала циркулювати велика кількість інформації. До якої мали доступ користувачі, які могли використовувати для різних цілей.

Розвиток баз даних та інформаційних систем привело до бурного розвитку високорівневих мов програмування в тому числі мов які використовуються для керуванням даними.

Тож, що таке база даних? Бази даних використовуються у випадках коли на потрібно зберігати чи обробляти, чи те, і інше великі об’єми даних. Дані зберігається та оброблюється в обчислювальних системах. Крім великої кількості інформації воно ще й повинна бути добре структурована. Тобто повинні бути виділені складові частини, та послідовні зв’язки між даними. Також структура баз даних повинна забезпечувати зручний пошук та обробку інформації.

Етапи розвитку архітектури баз даних:

1. Розвиток баз даних на великих ЕОМ, етап розробки перших СУБД.
2. Розробка СУБД для персональних комп’ютерів.
3. Розподілені бази даних.

Якщо говорити про бази даних на ЕОМ третього та четвертого покоління, то вони зберігали інформацію в звичайних файлах. Для програми такий файл - це іменована комірка в зовнішній пам’яті в яку можливо записати чи зчитати дані. Великим мінусом таких баз даних було те що структуру запису такого файлу знає лише прикладна програма яка з цим файлом працює. І в результаті отримуємо, що кожна програма повинна зберігати в собі структуру даних, яка

відповідає структурі цього файлу. Це призводило до того, що при спробі зміни структури даних файлу потребувалось перезаписувати, або створювати нові програми.

В епоху великих ЕОМ компанією IBM був створений дослідний проект, по дослідженню можливостей реляційної бази даних, і на основі цього дослідження була створена перша реляційна СУБД, яка отримала назву System R. За зразком цієї системи було створено інші, вже комерційні, СУБД, які також базуються на реляційній моделі даних. Популярність такої моделі ініціалізувала розробку високорівневих мов програмування для реляційної моделі.

На етапі розробки СУБД для персональних комп’ютерів, з'явилися програми, для не підготовлених користувачів. Такі програми були простими та інтуїтивним. До них відносились текстові редактори, електронні таблиці та інші всі вони мали інтуїтивний та зручний інтерфейс, який дозволяв користувачеві зручно обробляти інформацію та формувати звіти.

Поява таких програм вплинула і на базах даних, які були розроблені для персональних комп'ютерів і зберігати досить великі обсяги інформації. Тому персональні комп'ютери стали невід'ємним інструментом для виконання власних облікових функцій, звичайними користувачами. Доступність персональних комп'ютерів дала можливість користувачам, які раніше не використовували комп'ютери в своїй діяльності, розпочати використовувати їх. Зростання популярності розвинених і зручних програми по обробці даних змусив розробників програмного забезпечення розробляти і створювати нові та вдосконалювати старі системи. Такі системи називають настільними СУБД, або десктопними моделями.

Історія баз даних розвивається по спіралі, тому процес персоналізації баз даних пішов зворотний процес. Збільшуються кількість локальних мереж, розвиваються інтернет, і все більше даних розповсюджуються між комп'ютерами. Доступ до цих даних можуть мати різні користувачі для

використання їх в різних цілях. Тому гостро постає питання узгодженості даних, які використовуються різними користувачами. І успішне вирішення цих завдань привело в появі розподілених СУБД, які зберігають всі переваги існуючих раніше настільних СУБД, і в той же час дозволяють організовувати паралельну обробку інформації і забезпечують цілісності в базах даних.

Із швидким розвитком інформаційних систем, які базуються на даних, було створено окреме програмне забезпечення , яке відповідало за обробку та зберігання інформації. Розроблялися системи управління базами даних, або DBMS (data base management systems).

Перше, що призвело до їх появи стало, високий пріоритет даних в різних областях застосування. Для прикладу візьмемо банківську систему, яка зберігає інформацію про всі транзакції клієнта і ці данні повинні бути надійно захищені. Другою причиною появи СУБД можна вважати дублювання засобів управління даними при розробці різних додатків. Через це вартість розробки таких додатків була досить висока. Також був необхідний доступ до однієї і тієї ж інформації користувачам, які працювали з одними і тими ж даними в різних додатках.

Розвиток СУБД і поширенням персональних комп'ютерів відбувались одночасно. Це призвело до появи досить висока конкуренція на ринку СУБД, що змусило розробників удосконалювати ці СУБД, пропонуючи нові можливості, покращувати інтерфейси і пришвидшувати системи управління базами даних. Велика кількість СУБД, які виконували однакові функції, ініціювали появу різноманітних рішень для конвертації даних з однієї системи в іншу.

Такий етап можна охарактеризувати так: в цілому системи управління були розраховані на монопольний доступ, такий доступ передбачає виключно роботу одного користувача з інформаційною системою СУБД. Проте, існувала можливість послідовної роботи з даними для користувачі, виконуючи послідовно завдання. Наприклад, бухгалтер вводити дані для виконання

банківських операцій, і після того як бухгалтер закінчить роботу з інформаційною системою, головний бухгалтер зможе виконати безпосередньо банківські операції.

Велика кількість СУБД отримали розвинені та зручні інтерфейси-розробки додатків не залучаючи програмування. Інструменти розробки складалися із готових елементі, форм, звітів, графічних конструкторів запиту та і.н. Десктопні СУБД підтримували специфічні мов управління даними на рівні окремих рядків і таблиць. Особливістю тогочасних СУБД була відсутність підтримки цілісності і узгодженості даних, а потребувалось недопускання запису в бази даних некоректної інформації. Така перевірка виконувалась на стороні додатка, або цю функція перевірки даних перекладали на кінцевого користувача, який повинен був вводити в систему коректні дані та слідкувати за їх коректністю.

Прикладом тогочасних СУБД в першу чергу є сімейство dBase, FoxPro, системи Clipper, Paradox. Розглянемо ринок сучасних СУБД. Рейтинг сучасних СУБД можна створювати використовуючи різні критерії, можна використовувати дані опитування користувачів, в деяких випадках можна оцінювати кількість проданих примірників. У деяких випадках можна оцінювати по грошовому еквіваленту продажів. Але в кожному з рейтингу який-би ви не взяли, в кожному з них ви побачите в верхівці списку такі СУБД. Це СУБД Oracle, Microsoft SQL Server, DB2 (IBM), PostgreSQL та MySQL.

Коротко розглянемо окремо кожну з цих систем. Почнемо з системи під назвою DB2. Як описувалось раніше на початку 80-х років компанія IBM проводила дослідний проект з вивчення можливості реляційної баз даних. І на основі реляційної теорії була розроблена перша дослідницька СУБД яка отримала назву System R, або System Relational. System R була перейменована в DB2 у 1982 році, і до сьогодення ця СУБД є основним напрямком компанії IBM в області СУБД.

Наступною розглянемо СУБД Oracle від компанії Oracle. Oracle був заснований в 1977 році. І саме ця компанія випустила на ринок СУБД першу комерційно поширювану СУБД, так як вісі існуючі на той час СУБД були дослідними. Обсяг ринку СУБД становить на сьогодні близько 46 мільярдів доларів. Саме компанія Oracle займає близько 46% ринку сучасних СУБД.

Також потрібно згадати, ще SQL Server від Microsoft, який є лідером на ринку програмного забезпечення зі своєю операційною системою MS Windows та офісним пакетом MS Office. За основу взято програмний продукт, розроблений спільно з компанією Sybase. Перша СУБД, він цієї колаборації, побачила світ у 1988 року. SQL Server відрізняється від тих СУБД які описуються тут тим, що вони мульти-платформні на відмінно MicroSoft SQL Server випускається виключно для операційну систему Windows, і на ринку СУБД цієї операційної системі SQL Server займає 53% ринку, але на глобальному ринку займає менше 20%.

PostgreSQL - це безкоштовна СУБД з відкритим програмним кодом, розроблена каліфорнійським університеті Берклі. Сьогодні СУБД підтримується ентузіастів. На такій основі базується СУБД MySQL, тобто вона також вільно поширюється, але вона також підтримується компанією Oracle. Ця СУБД хороший вибір для розробки невеликих і середніх програм, і саме цю СУБД ми обрали для реалізації нашого продукту.

Розглянемо функції, які характерні для СУБД. По-перше це рішення для постійного або довготривалого зберігання інформації. Окрім цього, СУБД реалізують рівні доступу до інформації для користувачів, тобто частина даних може бути доступна тільки деяким користувачам, деякі користувачі можуть мати обмеження по редагуванню даних. СУБД реалізують захист, цілісність та узгодженості даних. Описуючи структуру даних, ми описуємо конкретні правила предметної області, яким дані повинні відповідати. Кожна СУБД дає можливість використовувати високорівневих і ефективних мов запитів. Тобто

ми можемо робити запити до інформації, не знаючи зовсім нічого про фізичне зберігання даних.

Важливою функцією СУБД є незалежність від фізичних та логічних рівнів зберігання даних. Всі додатки, які працюють з базою даних, не повинні залежати від того, як фізично зберігаються дані. В життєвому циклі СУБД може відбутися так, що потрібно перенести частину даних на інший пристрій зберігання або розбити якусь таблицю на частини, але це ніяк не повинно вплинути на роботу програми, яка працює цією інформацією. Також на протязі використання, СУБД може модернізовуватись структури. Так як не завжди можливо врахувати в повному обсязі особливості предметної області на першому етапі створення системи.

Розробка інформаційної системи починається із визначення їх архітектуру, розуміння, всіх ролей та функцій компонентів системи, з яких вона складатиметься, та взаємодіяти між різні компоненти. Також ми повинні визначити, яке обладнання потрібно для роботи нашої інформаційної системи, і визначитися всі допоміжні програмні засоби, необхідних для правильного функціонування системи. Наприклад, які програмні засоби нам можуть знадобиться? Перше це безумовно операційна система, також не обійдеться без деяких драйверів. Ще одним необхідним програмним засобом стане система управляння базами даних для зберігання даних. Можуть знадобитися спеціалізовані програмні пакети, реалізовані спеціально для нашої інформаційної системи і вирішальні функції її бізнес-логіки.

Програмні компоненти будуть виступати в таких ролях. Наприклад програма яка реалізовує взаємодію з користувачем, через яку користувач зможе отримувати доступ до інформаційної системи називається “Клієнт”. Функціональний модуль не від'ємна частина інформаційної системи, який реалізовує основний функціонал системи, та її бізнес-логіку. Компонент який

забезпечує зберігання і реалізує надійний доступ до даних, які потрібні для роботи інформаційної системи отримав назву “Сервер”.

Архітектур інформаційних систем можна поділяти на:

- однокомпонентні;
- компоненти «клієнт — сервер»;
- багатошарові архітектури;
- гібридні моделі.

Сервером називають комп’ютер, який керує тим чи іншим ресурсом. Комп'ютер, який бажає отримати доступ до ресурсів, прийнято називають клієнтом.

Додаток має 4 групи функцій:

- Функції вводу виводу даних;
- прикладні функції;
- фундаментальні функції зберігання та управління даними;
- службові функції.

В групу службових функції входять обов’язки зв'язку між функціями перших трьох груп.

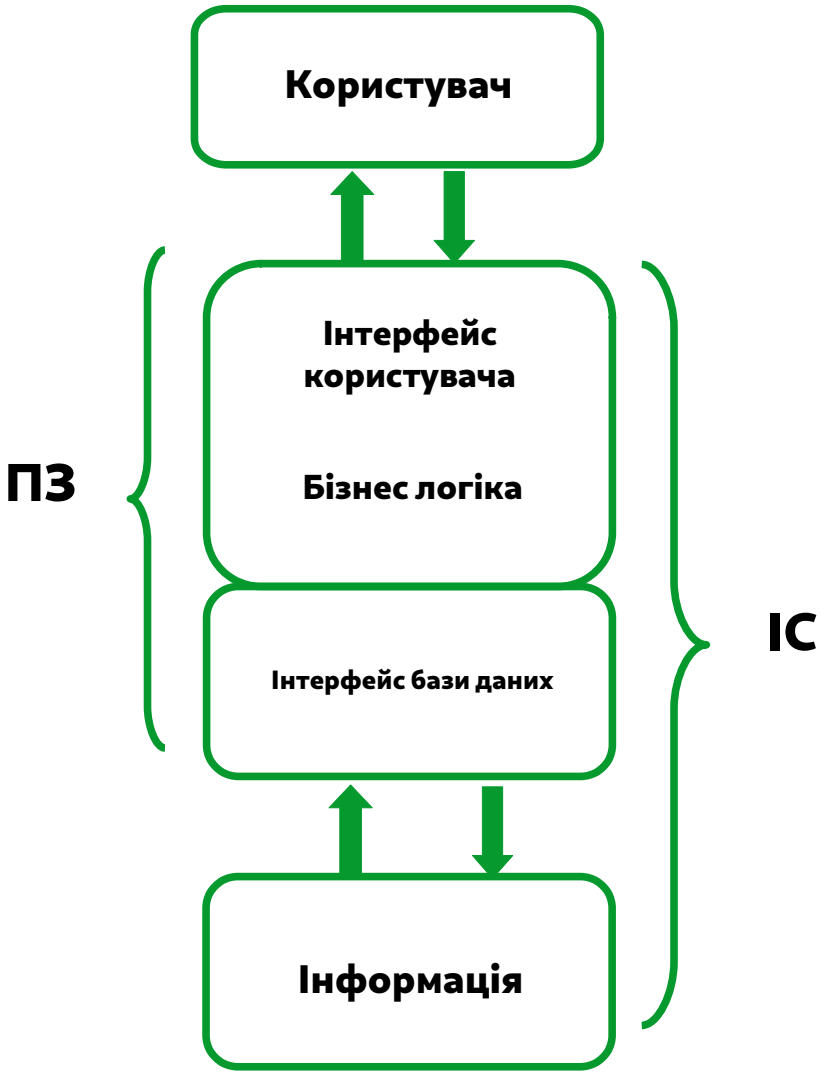

Рис. 2.1. Схема взаємодії логічних компонентів

В нашому додатку можна виділити наступні логічні компоненти. Це компонент взаємодії з користувачем, який буде реалізовувати функції першої групи, тобто інтерфейсні функції. Прикладні компоненти, які реалізують підтримувати функції другої групи. Та Компонент, який надає доступу до інформаційних ресурсів .

Під час вибору архітектури інформаційної системи нам потрібно розуміти, яке програмне забезпечення буде інтегроване в ті чи інші програмні компоненти, які механізми реалізовують функції програмного забезпечення, як розподілені логічні компоненти між комп'ютерами в мережі і яким чином компоненти зв’язуються між собою.

Однокомпонентну систему, використовується досить рідко. У систем такої архітектури всі програмні компоненти реалізовані майже завжди в одному програмному модулі, завдяки цьому спеціальний програмний код вбудований в базу даних, або, навпаки, база даних вбудована в якийсь додаток. Зазвичай, це системи розраховані на одного користувача.

Наступною розглянемо архітектура «клієнт — сервер». В такій архітектурі ролі клієнта і сервера даних виконуються на різних системах. Особливостями таких систем;

- постійне з'єднання між клієнтом і сервером;
- спільне використання даних;
- висока надійність системи для деякої кількості користувачів.

Виділяють чотири підходи до архітектури «клієнт - сервер»:

- модель файлового сервера (File server - FS);
- модель доступу до віддалених ресурсів(Remote Access Data - RDA);
- модель сервера бази даних(DataBase Server DBS);
- модель сервера додатків(Application Server - AS).

Коротко розглянемо особливість кожного підходу. Перший підхід - модель файлового сервера. При використані такого підходу компонент візуалізації та прикладний компонент реалізуються на стороні сервера, а компонент доступу до інформаційних ресурсів реалізується на сервері. Обмін даними між клієнтом і сервером відбувається за допомогою передачі фрагментів файлу. Таку архітектура реалізують тільки для інформаційних систем з невеликою кількістю користувачів. Основні мінуси такої архітектури:

- високий трафік, тому що ми передаємо запит у вигляді блоку файлів і отримуємо по мережі блоки файлів від сервера до клієнта;

- відсутні адекватні засобів захисту даних, так як захист відбувається на рівні файлової системи. Запис блоків файлів відстежує тільки клієнт.

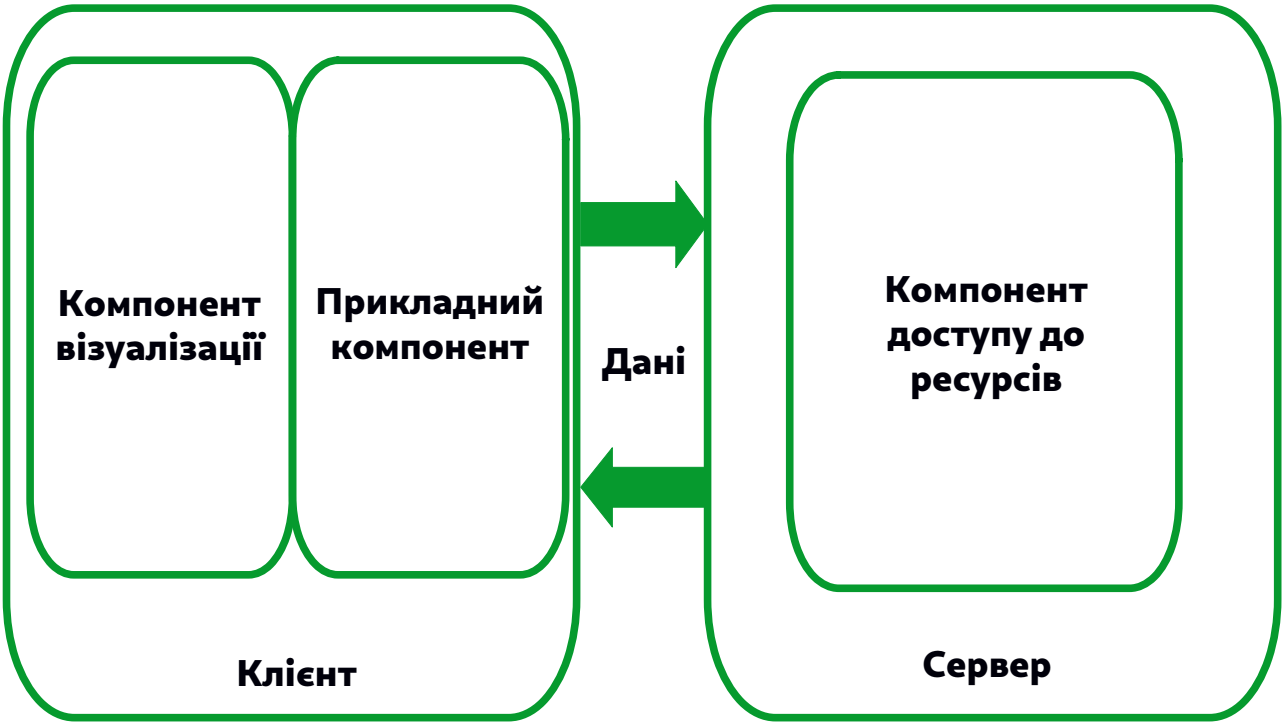

Рис. 2.2. Схема файлового сервера

Наступною розглянемо модель доступу до віддалених ресурсів така модель використовує такий же розподіл компонентів, але запити до ресурсів формується за допомогою спеціальної мови, який яка отримала назву SQL(*Structured query language — мова структурованих запитів*) які обробляються сервер. При такій моделі запити передавалися на уніфіковані мові SQL, яка підтримується будь-якою реляційної СУБД. Завдяки цьому трафік мережі знижувався, так як від сервера до клієнта передається тільки та інформація, яка була знайдена за допомогою запиту який оброблюється на стороні сервера. Але адміністрування такий програми дуже складне, так як на стороні клієнта знаходилися функції візуалізації та бізнес логіка.

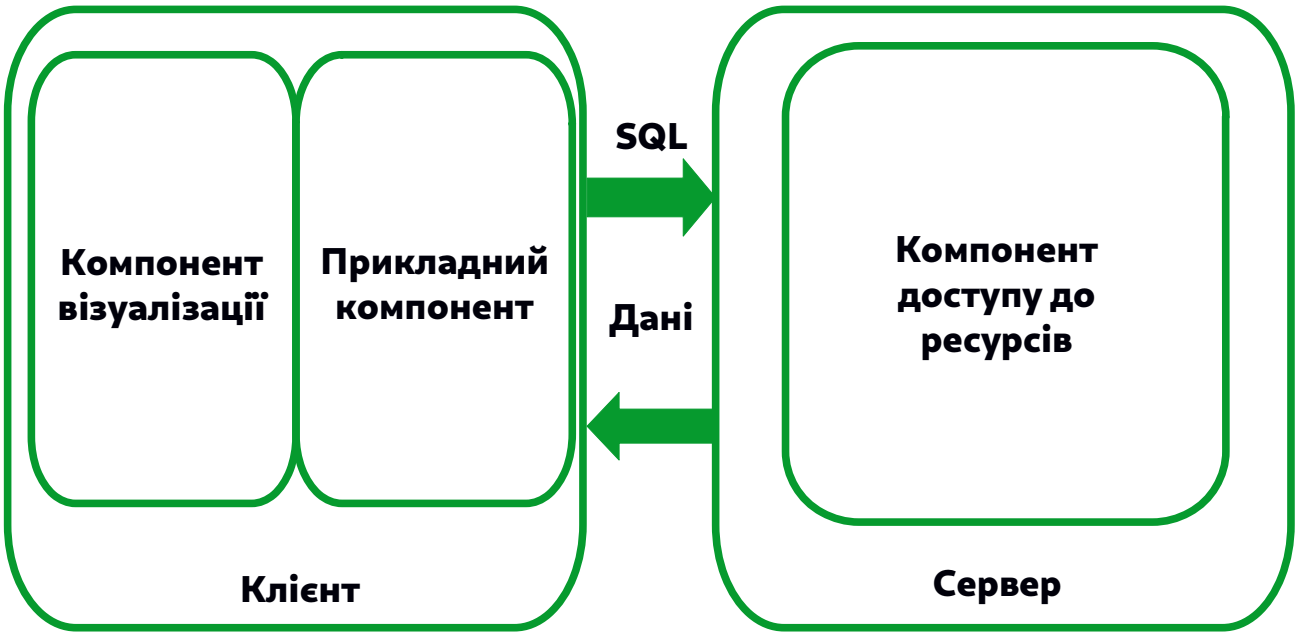

Рис. 2.3. Схема доступу до віддалених ресурсів

У ході вирішення цієї проблеми була розроблена модель сервер баз даних. При використані такої моделі вся бізнес логіка, перенесена на сторону сервера і зберігався разом з даними, а на стороні клієнта залишився тільки компонент візуалізації. Управління такими ресурсами реалізується механізмом збережених процедур. Процедури зберігаються в базі даних і виконуються на тому ж комп'ютері, на якому зберігається інформація. А користувачеві залишається лише передати на сервер назви процедур, які повинні виконатись, а також вказує їх параметри. Мова, на які реалізується збереження процедур, являє собою розширення мови SQL і є унікальним для кожної конкретної СУБД. До переваг такої модель можна віднести:

- можливість централізованого адміністрування прикладних функцій. Функції розробляються раз, зберігаються на сервері бази даних і можуть бути багаторазово викликані комп'ютерами клієнтів.
- зниження мережевого трафіку: ми передаємо не запитав, а тільки лише ім'я процедури і параметри.
- Економія ресурсів, поділивши процедури між декількома програмами, економимо тим самим ресурси комп'ютера.

До недоліки такої моделі можна віднести:

- обмежені засоби, які використовуються для збережених процедур, так як засоби високорівневих мов програмування значно ширше і багатше.
- Обмежена сфера використання процедурного розширення мови SQL обмежена конкретною СУБД, і не у всіх СУБД присутні можливості, наприклад, налагодження і тестування процедур.

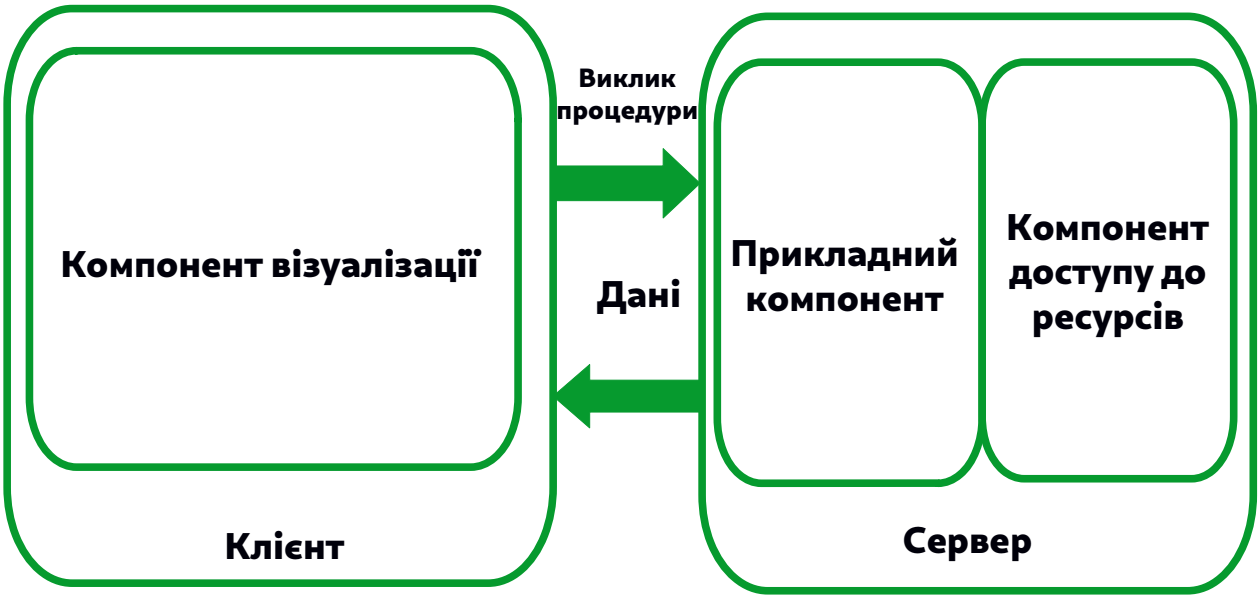

Рис. 2.4. Схема сервера бази даних

У деяких випадках поєднують модель віддаленого доступу до ресурсів і модель файлового сервера. В такому випадку прикладний частину, яка реалізує бізнес-логіку, поділяють між сервером та клієнтом. Частина клієнта знаходиться разом з функціями введення-виведення даних на комп'ютері клієнта. Вона містить найбільш складні функції, що реалізують потрібну бізнес-логіку, а серверна частина зберігається у вигляді процедури разом з сервером бази даних.

Результатом розвитку перераховних моделей стала трирівнева архітектура, де між клієнтом і сервером з'являється ще один рівень, на якому реалізується функції бізнес-логіки. Таку модель називають сервером додатків. Ця модель набула широкого розповсюдження в прикладних систем. Клієнтів такої системи поділяють на «тонких» і «товстих». «Тонкими» називають

клієнта, якому доступно лише функціонал введення і відображення даних. «Товстий» клієнт поєднує компонент візуалізації даних та прикладні компоненти.

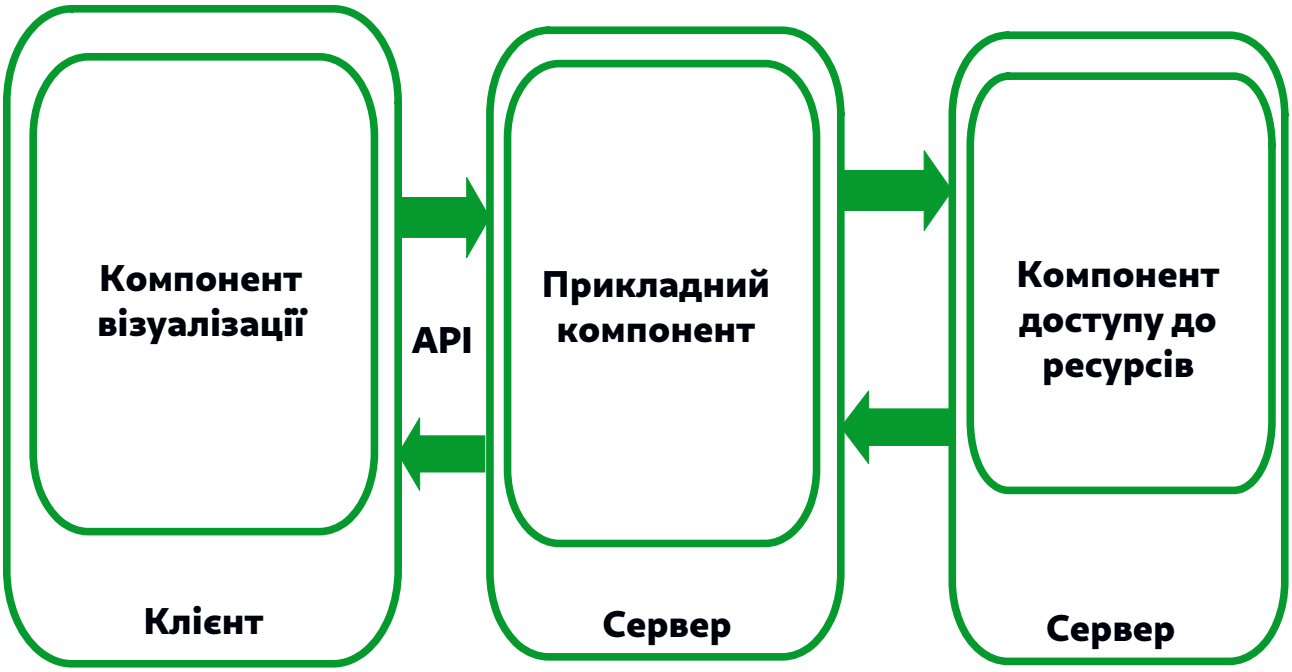

Рис. 2.5. Схема сервер додатка

У випадку коли для сервера додатка виділений в окремий рівень, то для його реалізації можуть використовуватись універсальні механізми багатозадачності операційної системи, а також стандартизовані інтерфейси між двома іншими компонентами. Декілька слів про участь сервера бази даних в такій системі. Потрібно, зберігалася не тільки інформація, але і деякі правила предметної області. Прикладом таких правил може бути, обмеження, які ми накладаємо внесення даних до бази. База даних повинна відповідати правилам предметної області, за якими вона функціонує. Також, необхідний постійно контролювати стан бази даних, необхідно відстежувати зміни і адекватно реагувати на них. Також необхідно, щоб виникнення певної ситуації в прикладній програмі впливало на хід виконання цієї програми. Виходячи з цього, можна сформулювати поняття активного сервера бази даних, який включає в себе процедури бази даних, правила, події в базі даних, типи даних, визначені користувачем.

Процедурами бази даних називають загальні частини прикладних програм оформлені в окремий програмний блок, який зберігається в базі данихДані разом з даними. Такі процедура можуть використовуватись різними прикладних програмами. Таким чином ми скорочуємо кількість прикладних програм, так як вони використовують готові процедури.

Для обробки ситуацій, які з'являються при зміні в базах даних, існує механізм правил, який дозволяє програмувати обробку ситуації. Правило встановлюється для таблиці бази даних і застосовується при виконанні над таблицею операції додавання, зміни або видалення даних. Використання цих правил виконується, як перевірка сформульованих в цьому правилі умов і виконання певних дій, які також в цьому правилі визначені. Такі правила зберігаються на сервері бази даних, і ніяк не залежить від прикладних програм.

Механізм подій в базі даних дає змогу одним програмами отримувати повідомлення про виконання певних дій, викликаних іншими програмами, завдяки цьому синхронізується робота різних додатків. Різні програми виконують дій в базі даних, а сервер сповіщає інші, зацікавлені в цих діях програми, про їх виконання. Реакція на виконання якоїсь дію може бути виконання певних запрограмованих розробником дій іншої програми.

Не завжди, для зручного опису даних, на достатньо стандартних типів даних, тому нам можуть знадобитися типи даних, визначені користувачем. Такі типи можливо створити майже в будь-який СУБД, нові типи даних створюються на основі існуючих, за допомогою опису з використанням різних обмежень. Також разом з даними ми можемо зберігати обмеження цілісності, або значення за замовчуванням, тобто правила. Отже, в цьому блоці ми розглянули різні архітектури інформаційних систем та обрали для свого додатку трирівневу архітектуру "Клієнт-Сервер" з моделлю сервер додатку.

2.2 Проектування бази даних

Розглянемо, з яких етапів складається створення бази даних. В першу чергу необхідно визначити предмету область, дані про яку ми будемо зберігати та специфікувати всі вимоги. Це може бути об'єкти реального світу, їх зв'язки, стосунки один з одним. Також необхідно заздалегідь визначити можливі розміри баз даних, яка кількість користувачів. Виходячи з цих вимог підбирається реальна СУБД, в якій будуть описуватися основні структури та об'єкти, необхідні для зберігання інформації. Також обрана СУБД повинна давати можливість реалізовувати правила цілісності та бізнес-логіку.

Після вибору СУБД, опису структури бази даних, можемо переходити до реалізації бази даних. В цьому етапі створюються об'єкти бази даних, які потім заповнюються даними. Потім база даних тестується, де перевіряється, що всі дані описані коректно, правила бізнес-логіки функціонують коректно.

Розглянемо перший етап процес моделювання. Моделювання потрібно починати з вивчення понять та опису. Ми обов'язково повинні розуміти, які дані ми будемо зберігати, і най важливіше, яка інформація повинна відповідати запитам користувача до бази даних. При моделюванні описуємо об'єкти та поняття вибраної предметної області та реалізуємо зв'язки між ними. Також, обов'язково потрібно описати обмеження цілісності, вимоги до допустимих значень і зв'язків між ними.

Розглянемо предметну область, інформаційної системи, яку ми створюємо «автоматизована система супроводу навчального процесу». Обов'язково потрібно зберігати інформацію, розподіл студентів по групам, зберігати інформацію про студентів. Скоріше за все, данні про студента буде містити, прізвище, ім'я, номер залікової книжки, і ми розуміємо, що в номері залікової книжки не повинно бути символів, прізвище та ім’я складається тільки із букв, студент повинен бути лише в одній групі. Потім вказуючи, конкретний тест, цей тест повинен бути по конкретному предмету, його проводить конкретний

викладач. Результатом тесту є оцінка. Таким чином, описали предметну область проведення тесті та обмежених цілісності, а також правила бізнес-логіки.

На етапі моделювання даних складається семантична модель. Це високорівнева модель, тобто потрібно абстрагується при складанні семантичної моделі від конкретної СУБД, в рамках якої буде реалізована база даних, також від особливостей фізичного зберігання Процес моделювання починається з аналізу предметної області та виявлення вимог користувачів. Якщо розглядати базу даних нашої системи, то різні користувачі будуть висувати різні вимоги, до бази даних. Студентам потрібно знати, коли і в якій аудиторії буде проводитись пара, викладач повинен мати можливість встановити оцінку, а адміністратор навчального закладу повинен створити розклад та визначити предмети та викладачів для проведення пар.

Представлення окремих користувачів узагальнюються в концептуальній схемі бази даних. Таку схему можна описати звичайними словами за допомогою математичних формул, таблиць, графіків та інших засобів, які допоможуть краще зрозуміти предметну область таку модель також називають інфологічна модель. Після розробки інфологічної моделі наступає етап логічного моделювання даних. В цій моделі ми описуємо структуру зберігання в обраній СУБД. Фізичну реалізацію зберігання даних виконують інструменти вибраної СУБД. Трьох-рівнева архітектура або сервер додатку забезпечує незалежність збереженої інформації від програм які їх використовують. Таким чином при необхідності можна перенести базу даних на інший пристрій, або розбити їх на частинах, при досягнені великих розмірів. Можна підключити до бази даних нових користувачів користувачів. Зміни в фізичній моделі ніяк не відображаються на логічній моделі даних. І навпаки, якщо потрібно відредагувати логічну структуру даних, то це не буде потребувати змін фізичних засобів.

Для семантичного моделювання найбільш поширеною моделлю є модель «сутність - зв'язок», запропонована Питером Ченом у 1976 році. Такі модель більшості випадків представляють у графічній формі з використанням нотацій Чена, яка називається діаграмою «сутність - зв'язок» або entity-relationship diagram. Основні переважні такої діаграми:

Рис. 2.6.Елементи діаграми "сутність-зв'язок"

- наочність;
- можливість проектування бази даних с великою кількістю об’єктів та атрибутів;
- реалізована в багатьох системах автоматизованого проектування баз даних

Пізніше, були розроблені подібні моделі, але основною перевагою такої та подібних моделей є наочність, так як малюнок завжди зрозуміліший ніж текстовий опис. Діаграма «сутність - зв'язок» є інструментом концептуального рівня проектування, і важливою особливістю, такої моделі, те що вона не містить операції і тому не може бути використаний безпосередньо. Діаграми «сутність — зв’язок» використовуються для проектування даних, дозволяють краще зрозуміти поняття предметної області і зв'язки між ними, при цьому має простий, зручний графічний інтерфейс, який описує структуру бази даних. Основні елементами діаграми «сутність — зв’язок»:

- сутність;
- атрибути;
- зв’язок.

Під сутністю розуміється будь-який об'єкт або поняття, інформацію про буде зберігатися. Властивості називається атрибутами, та сутності вступають у відносини з іншим, які називають зв’язками. На рис.2.6 позначені основні понять, які можна використовувати в діаграмі «сутність - зв'язок».

Елементи які оточені прямокутником це незалежних супутників,. Подвійним прямокутником будуть оточені залежні сутності, атрибути зображуються в овалах. Важливі атрибути, які називають ключами, оточують овалами і підкреслюють прямою лінією. Набори зв’язків вважається ромб, а лінії будуть пов’язувати сутності з їхніми атрибутами.

У нашій базі даних повинні зберігатися інформація про студентів, дані про пройдені тести по конкретних предметах. Студента можна охарактеризувати номером залікової книжки, прізвищем, іменем, датою народження. Студент може мати номер телефонів, всі студенти поділені на групи, для групи вказується тест по окремому примету, визначається викладач, за проходження тесту студенту виставляється оцінка.

Тут можна виділити сутності студента, викладача, предмети, тести. Атрибутами будуть ім'я, прізвище, номер залікової книжки, оцінка та і.н. Можна побачити обмеження в такій базі даних, наприклад, в імені не буває цифр, оцінка буде додатне число, студент знаходиться в своїй навчальній групі.

Такі діаграми використаються проектувальником або аналітиком бази даних, для кращого розуміння предметної області. Також діаграми використовуються для документації. Діаграми «сутність - зв'язок» передають логічну структуру бази даних і допомагають зрозуміти взаємодію інформації в базі даних користувачеві.

Розкладаючи модель «сутність - зв'язок», потрібно розуміти фундаментальне поняття, сутності. Почнемо з класичного визначення. Арістотель сказав: «Сутність - це одиниця, що володіє самостійністю, на відміну від її стану чи відносин, які можуть змінюватись і залежать від часу, місця і зв'язків з іншими сутностями». Під сутністю будемо розуміти, абстрактний або реальний предмет якогось виду, дані про який ми зберігаємо. Будь-який предмет або поняття: студенти, викладачі, автомобілі, книги, банківські операції - все, що завгодно.

Кожна сутність повинна мати унікальне ім'я, і в рамках бази даних до цього імені застосовуватися одна й та ж інтерпретація. Набір однакових або схожих сутностей утворює множини сутностей. Одна із важлива вимога теорії множин говорить про те, що множини не дублюються. Сутності повинні відрізнятися один від одного. Тобто нам потрібні якось ідентифікувати їх.

Ідентифікатори є властивостями сутностей, але крім ідентифікаторів у сутностей є й інші властивості, які отримали назву атрибути. Атрибут - це окрема характеристика об'єкта. Кожен атрибут в рамках сутності має унікальне ім'я та певний тип даних. Сутність може мати будь-якою кількістю атрибутів, які необхідні для її опису. Значення будь-якого атрибута повинно бути атомарним. На діаграмі, з'єднання сутність з її атрибутами зображується як показано на рис.2.7.

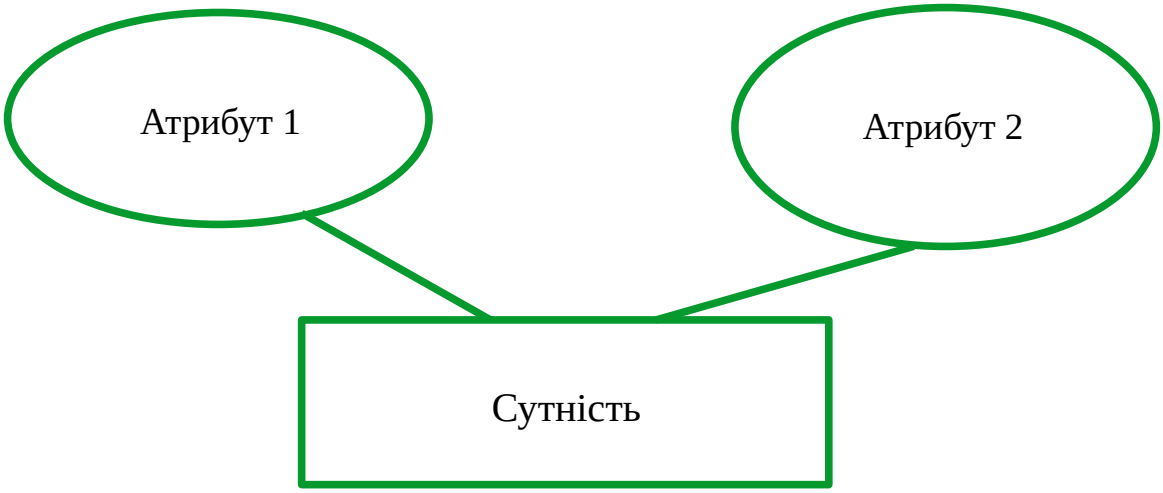

Рис. 2.7. З'єднання сутності з атрибутами

Значення атрибутів беруть з відповідної множини значень, визначеної для донного атрибута. При моделюванні виникає питання: як розглядати окремий елемент в якості атрибута сутності або в якості окремої сутністю? Прикладом може стати, адреса, якщо нас цікавить тільки місце прописки людини, то в такому випадку адреса буде атрибутом людини. А якщо нас цікавить адреса, як пункт призначення, то в такому випадку адреса буде, як окрема сутність.

Розглянемо як ідентифікувати сутність. Ідентифікація - це можливість

відрізняти одну сутність від іншої. Ідентифікація сутності відбувається за допомогою спеціального набору атрибутів, які називають ключами. Ключ - це один або декілька атрибутів об'єкта, який дозволяє однозначно відрізнити даний об'єкт від інших. У сутності студент ключем може бути номер залікової книжки так, як за номером залікової книжки можна однозначно визначити студента; може бути номер паспорта, або складна комбінація полів, наприклад, прізвище, ім'я, по батькові та дата народження. Не завжни вдається знайти природних ідентифікатор. Якщо не вдалось визначити такий природний ключ, то система може надати штучний ключ, який називається сурогатний ключ. Виділяють три типи ідентифікації:

- природні ключі;
- ключі «за положенням», наприклад, географічні, в порядку виникнення в часі;
- сурогатні, тобто штучні ключі.

Зручно, якщо кожний об'єкт має свій унікальний ідентифікатор, який не змінюється на протязі життєвого циклу об’єкта, і не змінюється в процесі роботи системи. Наприклад номер заліковки, номер паспорта. Такі ідентифікатори генеруються системою тому вони є сурогатними, але з точки зору людини можна використовувати як природній ключ. В цілому, не завжди вдається знайти такий атрибут або комбінацію атрибутів, для природного ідентифікатора, які дозволяють однозначно визначити об'єкт. Навіть якщо розглядати номер паспорта. Паспорта можна змінитися, залікову книжку теж можна загубити, тому дуже важко знайти такий ідентифікатор, який проживе весь життєвий цикл об'єкта, жодного разу не змінившись.

Що стосується ідентифікації за положення, то в сучасних інформаційних системах часто зберігаються дані, наприклад, що передаються з датчиків у вигляді простого потоку цифр, які можуть ідентифікуватися виключно

ідентифікатором датчика і якимось моментом у часі, в який дані з датчика були передані. Якщо серед атрибутів сутності вдається виділити декілька можливих ключів, то вибирається найбільш важливий, який буде найчастіше використовуватись при пошуку. Цей ключ називається первинним.

Тепер перейдемо до зв'язків. Сутності можуть взаємодіяти один з одним. Пошук одних сутностей за значеннями інших є одним із завдань бази даних. Для цього потрібно встановити зв'язки між сутностями. Складність побудованої моделі визначається саме наявністю цих зв'язків.

Зв'язок - це відношення між об'єктами. Також зв'язком можна вважати упорядкований набір сутностей. Зв'язків також мають ідентифікатори. Ідентифікатори зв'язку - це ідентифікатори сутностей, які вступають в зв’язок. На діаграмах «сутність - зв'язок» зв'язок зображується у виді ромбу, в якому зазначається ім'я для кожного зв'язку і з'єднуємо з сутностями, в залежності від типу зв'язку, направленими або ненаправленими лініями.

Критерії появи зв’язку:

- Коли необхідно зробити властивістю однієї сутності якусь іншу сутність або список сутностей
- коли потрібно записати в одну сутність ідентифікатор іншого.

Зв'язки, крім ідентифікаторів об'єктів, можуть мати власні атрибути. Подібні зв'язки об'єднуються в множини, так само як і сутності. Кардинальна відмінність сутностей і зв'язків полягає в тому, що об'єкти можуть жити своїм окремим життям незалежно від зв'язків, а зв’язки не можуть існувати окремо без об'єктів, яких вони пов'язують.

Зв'язку можна характеризувати трьома характеристиками:

- розмірність;
- потужність;

- модальність.

Розмірність зв'язку визначає кількість об'єктів, які пов’язує цей зв’язок. Виділяють наступні розмірності зв’язків:

- Бінарні;
- тернарні;
- n-арні;
- рекурсивні.

Найчастіше використовується зв'язок між двома об'єктів, вона називається бінарні. Але в деяких випадках потрібні більш складні зв’язки: наприклад, тернарні, якщо зв'язок утворюється трьома об'єктів; N-арні коли зв’язок утворюється з n об’єктами; також бувають рекурсивні зв'язку, коли в зв'язок вступають об'єкти одного і того ж виду. Прикладом таких зв'язків зображено на рис.2.8. Наприклад, сутність «студент» вступає в зв'язок з викладачем, здаючи

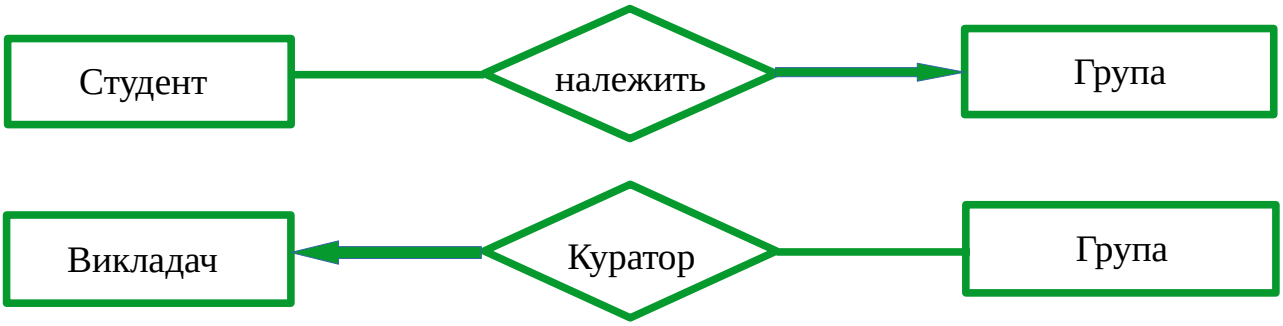

Рис. 2.8.Бінарні зв'язки

йому тест. Зв'язок між студентом і групою можна назвати як «зв'язок належить», також можливо вказати напрямок зв'язку від студента до групи, це вказує, що студент закріплений лише за однією групою. Між одними і тими ж об’єктами може бути більше одного зв'язку. Наприклад, студент може належати групі, а може бути старостою цієї групи.

Щоб утворився тернарний зв’язок потрібна участь трьох сутностей. Приклад тернарного зв'язку може стати складання тесту. В даному випадку сутностями будуть студент, який складає тест; предмет, по якому тест; і викладач, який цей і викладач який цей тест роздав. Також у нашого тернарного

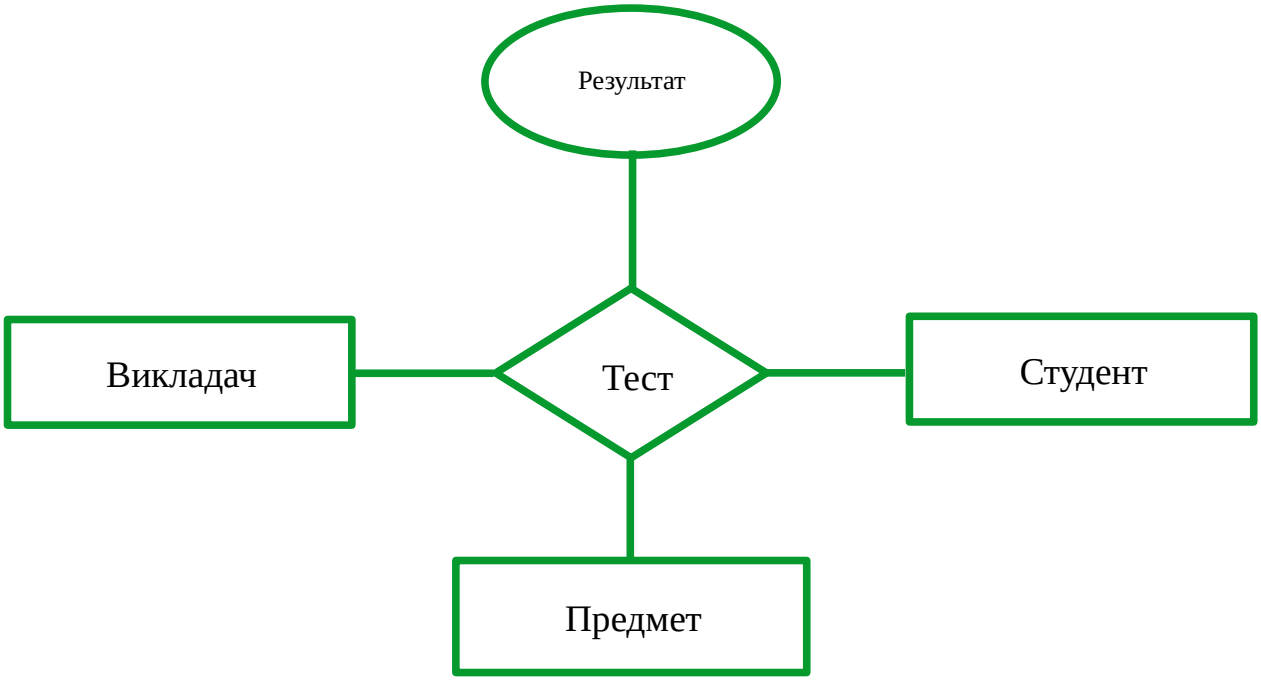

Рис. 2.9.Тернарний зв'язок

зв'язку буде свій власний атрибут — результат тесту. Атрибут “результат” належить зв’язку тому, що оцінка не є властивістю студента, не є властивістю предмета і не є властивістю викладача. Це буде саме властивість конкретної зв'язку, яку ми назвали тестом. В рамках моделі «сутність - зв'язок» також представляють і більш складні багатовимірні зв'язки. Але такі зв'язки складно реалізуються в контексті СУБД. Завжди можна перейти від багатовимірного зв'язку до бінарного. В наведеному прикладі використовується тернарний зв'язок “тест”, замінити такий зв’язок можна, введенням множин сутностей які з’єднуються. Тобто замінити зв'язок “тест” на сутність “тест”. І ця сутність буде вступати в бінарні зв'язки з іншими сутностями - зі студентом; з викладачем; і предметом. Таким чином, можливо перейти від багатовимірних зв'язків до бінарним рис.2.10.

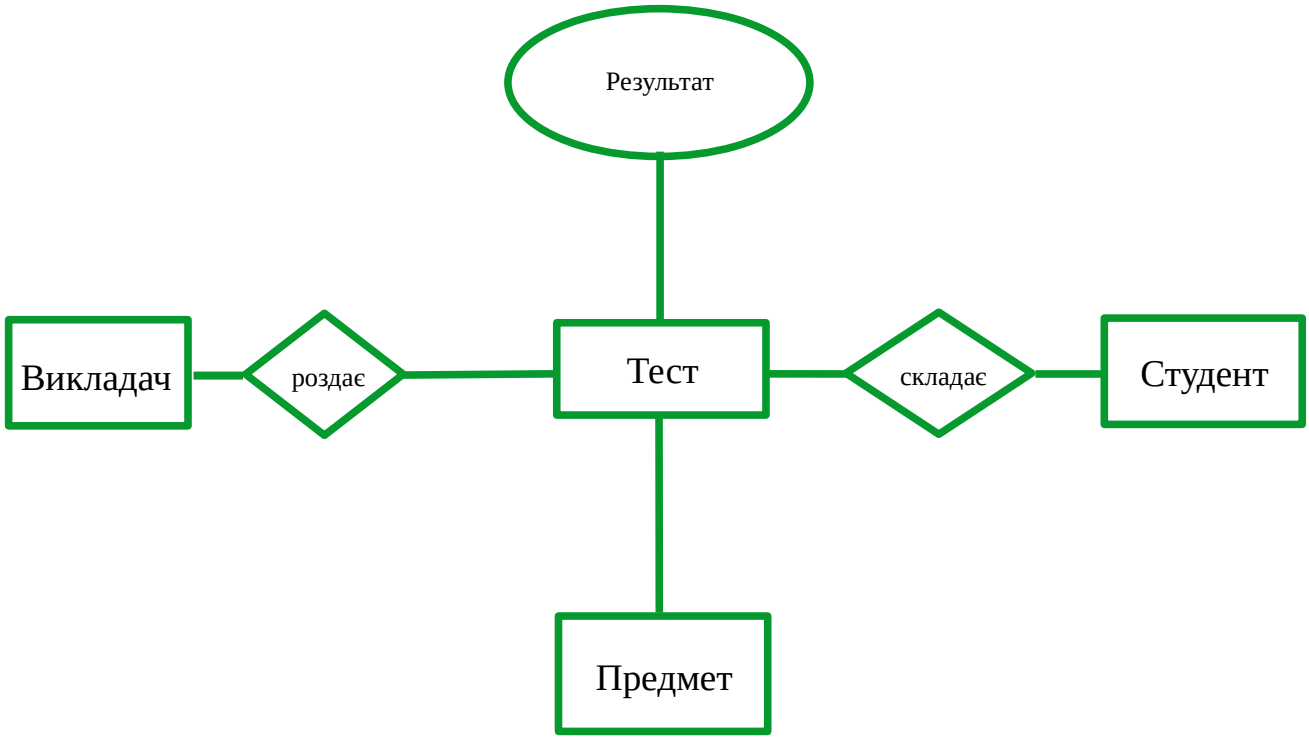

Рис. 2.10. Перехід від тернарного до бінарного зв'язку

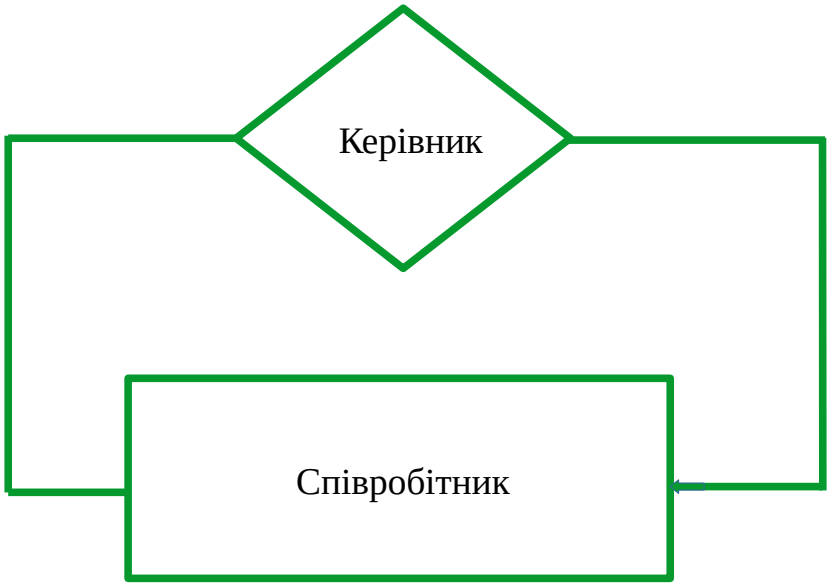

Рис. 2.11. Рекурсивний зв'язок

Розглянемо рекурсивний зв’язок. Припустимо, що потрібно зберігати дані про співробітників, і один із співробітників є керівником. Використовуючи рекурсивний зв'язок, ми можемо будувати складні ієрархічні структури рис.2.11.

Потужність зв’язків діляться на три види:

- “один-до-одного” 1:1;
- “один-до-багатьох”
- “багато-до-багатьох”

Зв’язок один до одного означає, що для кожного окремого екземпляру одної сутності відповідає один і лише один екземпляр другої сутності рис.2.12.

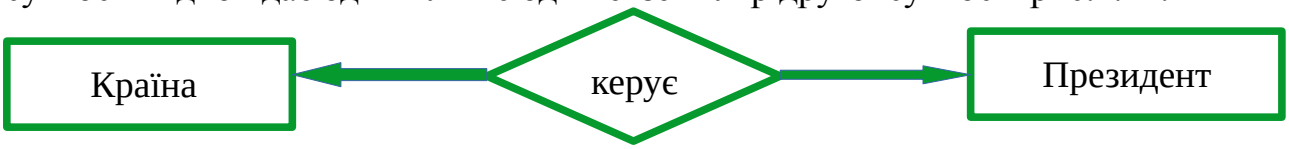

Рис. 2.12. Зв'язок один до одного

Наступний зв’язок один до багатьох, в такому зв’язку об’єкт одного виду може відповідати декілька об’єктів іншого виду, але не навпаки. Такий зв’язок показується як зображено на рис.2.13, де зв’язок з’єднується лінією в сторону до багатьох і стрілкою в сторону одного.

Рис. 2.13. Зв'язок один до багатьох

Зв’язок багато до багатьох найчастіше зустрічається, в такому випадку кожному об’єкту першої сутності відповідає декілька об’єктів другої сутності і навпаки рис.2.14.

Рис. 2.14. Зв'язок багато до багатьох

Створимо er-діаграму для нашої бази даних. Розпочнемо з виділення сутностей. Будуть наступні сутності: користувач з атрибутами логін та пароль, викладач з атрибутами ім’я, прізвище та супер-користувач, студент матиме

атрибути ім’я, прізвище, номер залікової книжки та стать, тест має атрибути питання, відповіді, назва та доступ і предмет має атрибут назва. Для простоти побудови такої діаграми використає web-сервіс Creately (creately.com). Цей сервіс надає зручний інтерфейс для створення різних діаграм. На рис.2.15 зображено er-модель нашої бази даних.

Розглянемо зв’язки, які зображені на er — моделі. Зв’язок “є” пов’язує сутність “користувач” та сутності “викладач” або “студент”. Цей зв’язок один до одного, так як один об’єкт сутності “користувач” відповідає одному і лише одному об’єкту сутності “викладач” або “студент”, і навпаки. Зв’язок “куратор” поєднує сутності “викладач” та “студент”. Такий зв’язок має потужність один до багатьох, тому що викладач може бути куратором для багатьох студентів, але для студента може бути лише один куратор. Наступний зв’язок “створює” він поєднує сутність “тест” з сутністю “викладач” він так само, як і попередній зв’язок один до багатьох. Викладач може створити безліч тестів, і кожний тест буде створений тільки одним викладачем. Зв’язок “з” з’єднує “предмет” та “тест”, цей зв’язок один до багатьох, тому що може бути безліч тестів з одного предмету, але кожен тест тільки з одного предмету. Останній бінарний зв’язок “складає” з’єднує “студент” та “ тест”, цей зв’язок багато до багатьох. “Студент” може складати декілька тестів і один “тест” може складати декілька “студентів”. Зв’язок “викладає” є тернарним, тому що він зв’язує три сутності “викладач”, який викладає “предмет” для “студента”.

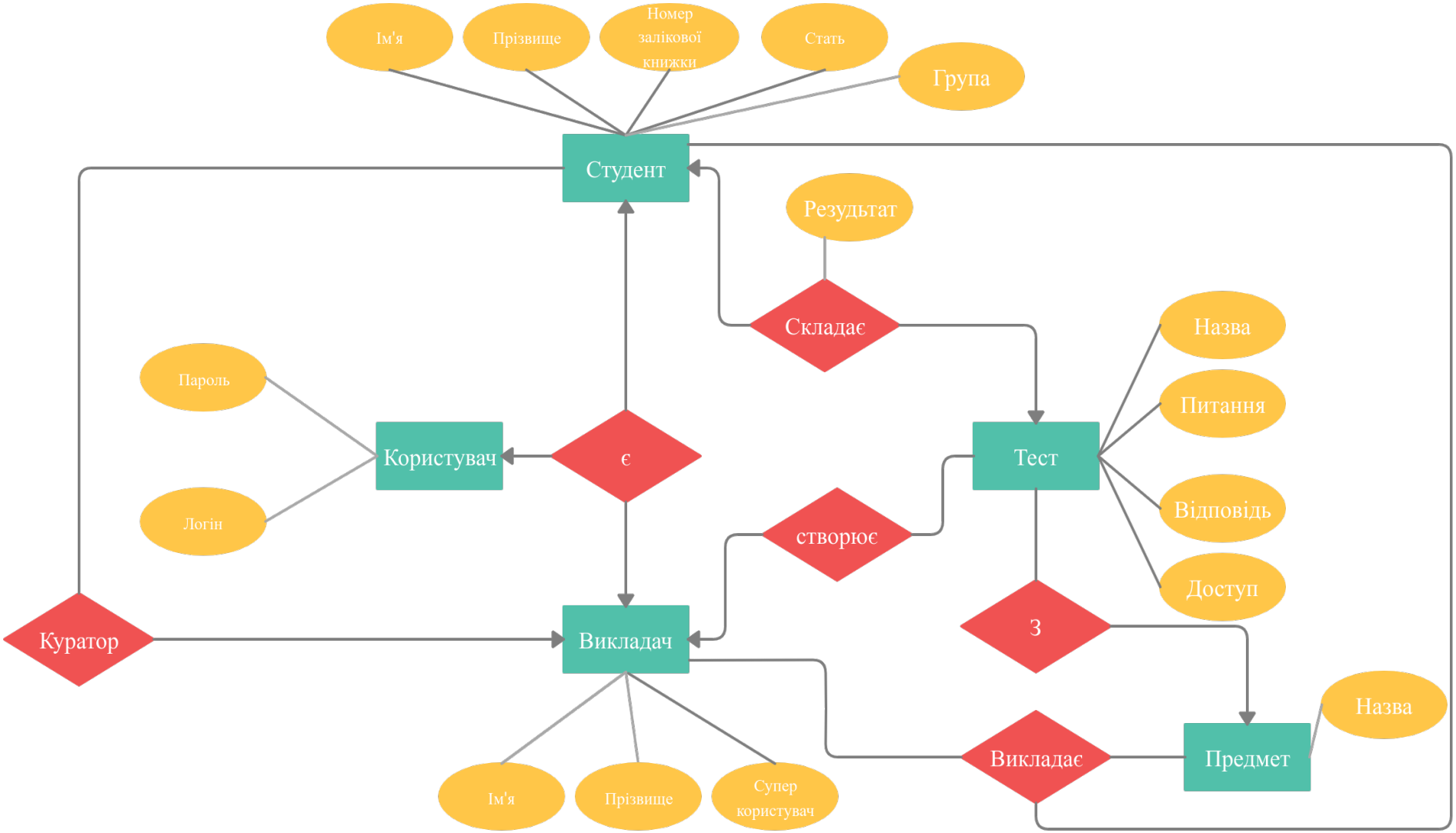

Рис. 2.15. ER -модель бази даних

Отже в цьому розділі ми розглянули види баз даних, які бувають СУБД, моделі реляційних баз даних, та побачили, як моделювати базу даних. Роблячи висновок з цього розділу ми обрали реляційну модель бази даних та обрали архітектуру сервер додатку для нашої інформаційної системи. Також ми обрали СУБД MySQL, тому що воно безкоштовне, має зручний інтерфейс з додатком MySQL Workbrench, та досить широкий інструментарій для створення та роботи з базами даних. І в кінці створили er - модель нашої бази даних рис.2.15.

Розділ 3 розробка автоматизованої системи супроводу навчального процесу

3.1 Реалізація бази даних

Реалізацію бази даних почнемо з розробки EER моделі. EER — це модель високого рівня або концептуальна модель даних, що включає розширення до оригінальної ER-моделі, що використовується при проектуванні баз даних. Інструменти MySQL Workbrench дозволяють з легкістю створити таку діаграму рис.3.1.

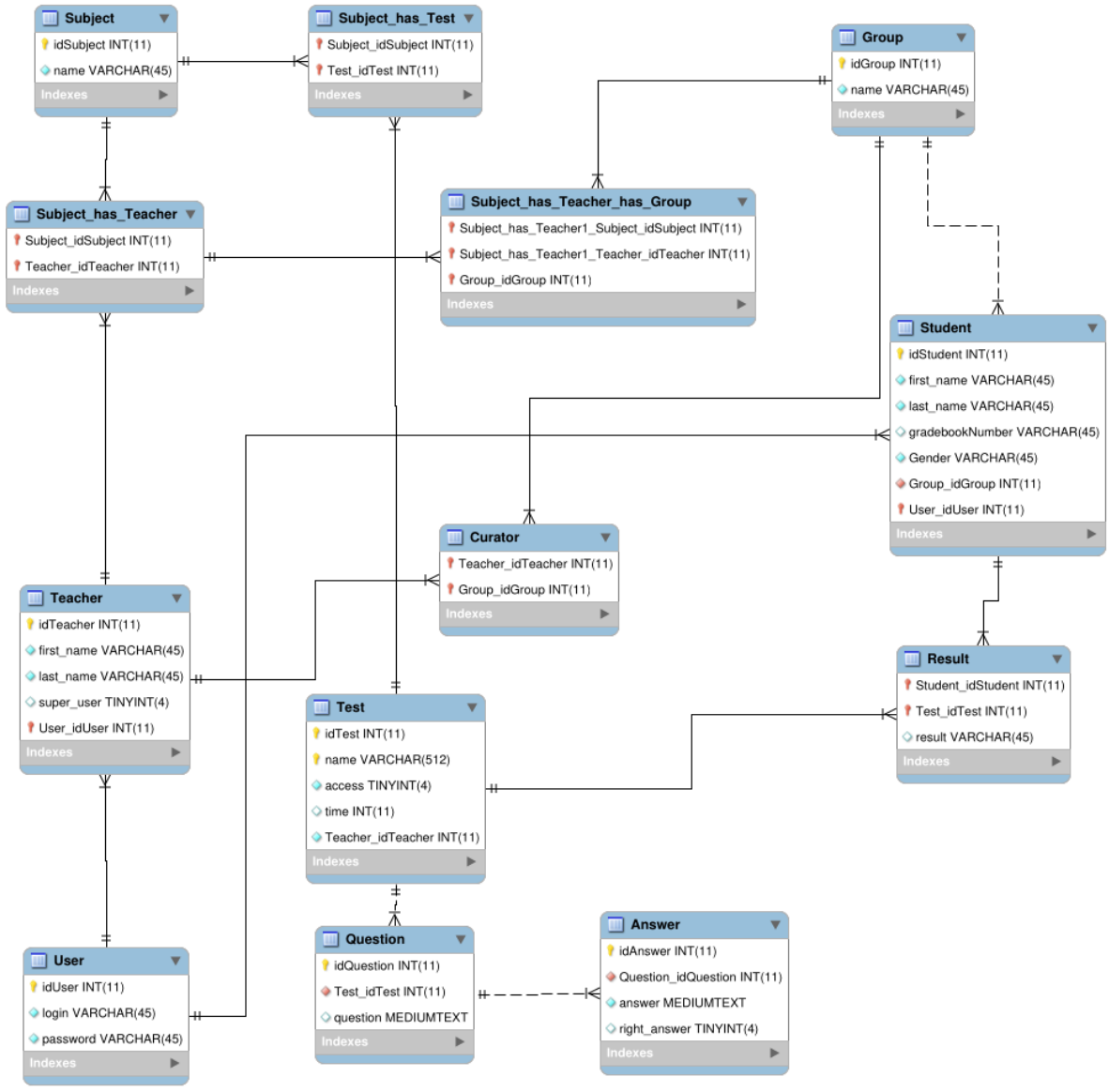

Рис. 3.1. EER діаграма

Модель на рис.2.3.1 показує всі сутності та атрибути у вигляді таблиць, тобто кожен елемент діаграми являє собою таблицю в базі даних. З першого погляду на діаграмі не відображені зв’язки, але це не так. Наприклад зв’язок “є”, щоб реалізувати таки зв’язок в реляційній базі даних потрібно додати атрибут до сутностей “студент” та “викладач”, який буде ідентифікувати, що конкретний “користувач” є конкретним “студентом” чи “викладачем”. Тому сутність “користувач” отримав атрибут “idUser”. Цей атрибут приймає унікальне значення при додавані нового користувача в базу даних. А сутності “студент” та “викладач” отримали атрибут “User_idUser”, цей атрибут є зовнішнім ключем та приймає значення атрибута “idUser” відповідного користувача.

Для реалізації сутності “тест”, ми “тест” розділили на 3 сутності: “Test” з атрибутами “idTest” для ідентифікації, “name” - назва, “access” — доступ, “час” для зберігання інформації на протязі якого часу цей тест проводиться та “Teachet_idTeacher”, що реалізує зв’язок “створює” рис.2.15. Сутність “Question” — питання з атрибутами “idQuestion” для ідентифікації, “Test_idTest” реалізує зв’язок між “Test” та “Question”, “question” де зберігається саме питання. І сутність “Answer” з атрибутами “idAnswer” для ідентифікації, “Question_idQiestion” для зв’язку з “Question”, “answer” для збереження відповіді, “right_answer”, що вказує чи є ця відповідь правильною.

Сутність “студент” рис.2.15 отримала такі ж атрибути окрім “групи” та додаткового атрибута “User_idUser” для реалізації зв’язку “є”. Атрибут “група” був перетворений в окрему сутність “Group”, для реалізації цілісності, обмежень і полегшення доступу та зміни. Тому в “Student” атрибут “група” замінено на “Group_idGroup”. Сутності такі як “Group” називають довідниками і зазвичай вони мають тільки два атрибута ідентифікатор та назву. Зв’язок “складає” рис.2.15 реалізований окремою сутністю “Result” рис.3.1. Ця сутність є залежною так, як вона не може існувати без двох інших. Тому “Result” має два

зовнішніх ключа “Student_idStudent”, що ідентифікує “Student” та “Test_idTest”, що ідентифікує тест та атрибут самого результату “result”.

Сутність “Teacher” рис.3.1 також отримав такі ж атрибути, як і “викладач” на er — моделі рис.2.15 окрім доданих атрибутів ідентифікації “idTeachet” та “User_idUser”. Так як викладач може бути куратором для групи студентів, для цього реалізована сутність “куратор”, яка має атрибути “Teacher_idTeachet”, який ідентифікує викладача, та “Group_idGroup”, що ідентифікує групу.

“Предмет” рис.2.15 реалізується “Subject” рис.3.1 і також має атрибут “name” -назва, а також ідентифікатор. Зв’язок “з” рис.2.15 реалізується залежною сутністю “Subject_has_Test”, що зберігає ідентифікатори “Test” та “Subject”. Зв’язок “викладає” рис.2.15 є тернарним і як описувалось раніше такий зв’язок досить складно реалізовувати в реляційних базах даних. Тому для реалізації такого зв’язку знадобилось створити дві залежні сутності “Subject_has_Teacher” та “Subject_has_Teacher_has_Group”. “Subject_has_Teacher” зберігає, які предмети викладає викладач, для цього потрібно два атрибута, два ідентифікатора “Subject_idSubject” та “Teacher_idTeacher”. “Subject_has_Teacher_has_Group” зберігає, який викладач викладає конкретний предмет для групи. Для цього ця сутність отримала атрибути “Subject_has_Teacher_Subject_idSubject” отримує значення “Subject_idSubject”, “Subject_has_Teacher_Teacher_idTeacher” отримує значення “Teacher_idTeacher” та “Group_idGroup” отримує значення “idGroup”.

Перейдемо від EER-моделі до таблиць для цього MySQL Workbrench дозволяє згенерувати SQL код. Розглянемо код для створення таблиці “User”.

```
CREATE TABLE IF NOT EXISTS `Tester`.`User` (
 `idUser` INT(11) NOT NULL AUTO_INCREMENT,
 `login` VARCHAR(45) CHARACTER SET 'ascii' COLLATE 'ascii_bin' NOT NULL,
 `password` VARCHAR(45) CHARACTER SET 'ascii' COLLATE 'ascii_bin' NOT NULL,
 PRIMARY KEY (`idUser`),
```

```
  UNIQUE INDEX `login_UNIQUE` (`login` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 215
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_unicode_ci;
```

CREATE TABLE IF NOT EXISTS `Tester`.`User` - інструкція для створення таблиці “User” в базі даних “Tester”, якщо такої такої нема. Далі в дужках вказуються атрибути та їх властивості. NOT NULL вказує нате, що атрибут обов’язково повинен бути визначений. Також вказується який тип даних має атрибут, наприклад ‘idUser’ має тип int, тому що він зберігає порядковий номер об’єкта сутності і являється первинним ключем, на це вказує PRIMARY KEY (`idUser`). ‘login’ та ‘password’ можуть мати символи, тому вони отримали символьний тип даних. ‘login’ також отримав властивість унікальності, для забезпечення цілісності то захищеності даних. Унікальність реалізується за допомогою інструкції UNIQUE INDEX `login_UNIQUE` (`login` ASC)). Інструкція ENGINE = InnoDB вказує на механізм зберігання даних. В нашому випадку це InnoDB, він використовується за замовченням в СУБД MySQL. DEFAULT CHARACTER SET = utf8mb4 описує тип кодування для зберігання символьних даних. COLLATE = utf8mb4_unicode_ci вказує на правила порівняння символьних даних. Весь SQL код вложено в додаток 1.

Отже в цьому розділі за допомогою MySQL Workbrench створили розширену модель сутність-зв’язок. Розглянули як реалізуються різні зв’язки в реляційній базі даних. Розглянули процес конвертації EER-моделі в таблиці SQL.

3.2 Сервер додатку: сервер

Для реалізації архітектури сервер додатку рис.2.5 потрібно 3 модулі: клієнт, сервер та сервер бази даних. Сервер бази даних реалізується завдяки потужностям СУБД MySQL, для якої була створена база даних. Розглянемо сервер, для написання сервера було обрано мову програмування Java.

Доступ до бази даних реалізується за допомогою модуля mysql-connector. Було створено клас DatabaseHandler, для реалізації функціоналу зберігання, завантаження, обновлення та видалення даних з бази даних. Також два допоміжні класи Configs та Constant.

Configs зберігає дані про базу даних, такі як хост бази даних, порт підключення, ім’я користувача під яким сервер звертається до бази даних, назва бази даних та пароль бази даних. Constant зберігає класи з назвами всіх таблиць та атрибутів, для зручності створення SQL запитів.

Для роботи з базою даних сервер відправляє SQL запити до СУБД, де вони обробляються і повертають результат запиту. Розглянемо функції які працюють з запитами. Функція для збереження даних про студента:

```
public boolean saveStudent(Student student) {
   String insert = "INSERT INTO " + Constant.User.USER_TABLE + "(" + Constant.User.LOGIN + "," +
Constant.User.PASSWORD + ")" + "VALUES(?,?)";

   try {
      PreparedStatement prSt = getDbConnection().prepareStatement(insert);
      prSt.setString(1, student.getLogin());
      prSt.setString(2, student.getPassword());

      prSt.executeUpdate();

      try {
         insert = "INSERT INTO " + Constant.Group.GROUP_TABLE + "(" + Constant.Group.NAME + ")" +
"VALUES(?);";
         prSt = getDbConnection().prepareStatement(insert);
         prSt.setString(1, student.getGroup());

         prSt.executeUpdate();
      } catch (SQLException | ClassNotFoundException e) {
         System.out.println("such a group already exists");
      }
```

```
 insert = "INSERT INTO " + Constant.Student.STUDENT_TABLE + "(" + Constant.Student.FIRST_NAME + ","
+ Constant.Student.LAST_NAME + "," + Constant.Student.GRADEBOOKNUMBER + "," +
Constant.Student.GENDER + "," + Constant.Student.idGROUP + "," + Constant.Student.USERID + ")" +
"SELECT ?,?,?,?," + Constant.Group.GROUP_TABLE + "." + Constant.Group.idGroup + "," +
Constant.User.USER_TABLE + "." + Constant.User.idUser + " FROM " + Constant.User.USER_TABLE + "
INNER JOIN Tester." + Constant.Group.GROUP_TABLE + " on " + Constant.User.LOGIN + " =? " + "
WHERE " + Constant.Group.NAME + " =? ";

        prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, student.getFirstName());
        prSt.setString(2, student.getLastName());
        prSt.setString(3, student.getGradebookNumber());
        prSt.setString(4, student.getGender());
        prSt.setString(5, student.getLogin());
        prSt.setString(6, student.getGroup());

        prSt.executeUpdate();
        dbConnection.close();
        return true;
    } catch (SQLException | ClassNotFoundException e) {
        updateStudent(student);
        return false;
    }
}
```

Для створення запиту оголошується змінна insert типу String. Insert ініціалізується "INSERT INTO " + Constant.User.*USER_TABLE* + "(" + Constant.User.*LOGIN* + "," + Constant.User.*PASSWORD* + ")" + "VALUES(?,?)", де INSERT INTO це ключові слова SQL, що виконують вставку даних в таблицю. Назва таблиці вказується далі, в даному випадку назву беремо із класу Constant. Далі в дужках вказуються атрибути які будуть записуватись. Їх назви також беремо з Constant. Після переліку всіх атрибутів закриваємо дужку та вказуємо ключове слово VALUES і одразу після нього в дужках вказуємо значення в тому ж порядку, що і раніше зазначені атрибути. Якщо ми хочемо вставити зміні то пишемо символ “?”, як показано в нашому прикладі. Далі використовуючи модуль sql-connector ініціалізуємо зміну prSt типу PreparedStatemen за допомогою функцій getDbConnection(). prepareStatement(insert), де ми передаємо в якості аргументу зміну insert. Далі замінюємо “?” на значення. Для цього використовуємо метод setString, де в якості аргументу передається два значення пере значення порядковий номер “?”,

значення, отримуємо наступну конструкцію prSt.setString(1, student.getLogin());. На даному етапі в нас готовий запит залишається його відправити до СУБД. Для відправлення запитів використовується метод executeUpdate(), отримуємо таку інструкцію: prSt.executeUpdate();. Якщо вісі дані вказано вірно, то в таблиці з'явиться новий запис. Таким чином створено запит на додавання користувача до бази даних. Далі додаються дані про групу, якщо такі та не існувала. SQL запит для запису інформації про групу створюється аналогічно. Далі створюємо запит на збереження інформації про студента. Цей запит дещо складніший, адже потрібно реалізувати зв'язок між користувачем, студентом та групою. Початок запиту реалізується так само з INSET INTO, назви таблиці, в дужках перелік атрибуті та вказуємо символи "?", для атрибутів "User_idUser" та "Group_idGroup", які реалізують зв'язки, дані потрібно брати з інших таблиць. Тому щоб отримати ці дані потрібно розширити запит. Ключове слово SELECT виконується, як вибір атрибутів, які вказуються після нього, далі вказується FROM, що вказує з якої таблиці брати атрибути назва якої вказується після ключового слова. Для того, щоб взяти потрібні дані з таблиці нам потрібно об'єднати таблиці, це можна зробити за допомогою інструкції INNER JOIN. Потім вказуємо таблицю, яку об'єднуємо, далі після ON вказуємо за яким параметром об'єднуємо таблиці, і в кінці запиту робимо вибірку за критерієм за допомогою WHERE та умовою після нього. Потім йде процес заміни "?". В кінці отримуємо такий запит:

```
INSERT INTO Student (first_name, last_name, gradebookNumber, Gender, Group_idGroup, User_idUser)
SELECT 'name', 'soname', 'gNumb', 'gender' , Group.idGroup, User.idUser FROM User
INNER JOIN Group ON login ='LOGIN'
WHERE Group.name = "group";
```

Для викладача запит створюється аналогічно. Далі розглянемо запит на завантаження даних, для цього використовується функція loadStudent(User).

```
public Student loadStudent(User user) {
    ResultSet resultSet = null;
    Student student = null;
```

```
 String select = "SELECT " + Constant.Student.STUDENT_TABLE + "." + Constant.Student.FIRST_NAME +
"," + Constant.Student.STUDENT_TABLE + "." + Constant.Student.LAST_NAME + "," +
Constant.Student.STUDENT_TABLE + "." + Constant.Student.GRADEBOOKNUMBER + "," +
Constant.Student.STUDENT_TABLE + "." + Constant.Student.GENDER + " FROM " +
Constant.Student.STUDENT_TABLE +
        " INNER JOIN " + Constant.User.USER_TABLE +
        " ON " + Constant.Student.STUDENT_TABLE + "." + Constant.Student.USERID + " = " +
Constant.User.USER_TABLE + "." + Constant.User.idUser +
        " WHERE BINARY " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =? and " +
Constant.User.USER_TABLE + "." + Constant.User.PASSWORD + " =?;";
   try {
      PreparedStatement prSt = getDbConnection().prepareStatement(select);
      prSt.setString(1, user.getLogin());
      prSt.setString(2, user.getPassword());


      resultSet = prSt.executeQuery();
      while (resultSet.next()) {
         student = new Student();
         student.setFirstName(resultSet.getString(Constant.Student.FIRST_NAME));
         student.setLastName(resultSet.getString(Constant.Student.LAST_NAME));
         student.setGradebookNumber(resultSet.getString(Constant.Student.GRADEBOOKNUMBER));
         student.setGender(resultSet.getString(Constant.Student.GENDER));
         student.setLogin(user.getLogin());
         student.setPassword(user.getPassword());
         student.setGroup(loadGroup(student));
      }
      dbConnection.close();
   } catch (SQLException | ClassNotFoundException e) {
      e.printStackTrace();
   }
   return student
}
```

LoadSudetn приймає аргумент типу User. Об’єкти типу User зберігають логін та пароль. Перейдемо до створення запиту. Нам так само потрібно зміна строкового типу даних, яка отримала назву select. Select ініціалізуємо фреймом

запиту. З початку вказуємо ключове слово SELCT, яке виконує вибір атрибутів, які ми хочемо вивести. Потім вказуємо всі атрибути, які нам потрібно взяти з бази даних та після FROM вказуємо з якої таблиці брати данні. Так як, дані про студента зберігаються, що найменше в трьох таблицях нам потрібно об'єднати деякі таблиці. Так наприклад, щоб отримати дані про студента маючи логін та пароль, потрібно об'єднати таблиці "Student" та "User" за допомогою INNER JOIN. І в кінці робимо вибірку за параметрами, але в даному випадку присутнє ключове слово BINARY, що враховує регістри при порівнянні символів. Також в SQL мові, які в багатьох інших мова присутні логічні оператори, тому для правильної аутентифікації використовуємо логічний оператор AND. Потім зміну insert передаємо на заміну "?". До СУБД відправляється наступний запит:

```
SELECT first_name, last_name, gradebookNumber, Gender FROM Student
INNER JOIN User ON Student.User_idUser = User.idUser
WHERE BINARY User.login ='login' and User.password ='password';
```

Після отримання готового запиту, відправляємо його до СУБД, за допомогою методу executeQuery(), цей метод повертає результат запиту, який ми відправляли. Для обробки даних, які повернулись потрібно сотворити зміну типу ResultSet, яку назвали resultSet. Результат методу executeQuery() присвоюємо resultSet. Далі працюємо з цією змінною, створюємо цикл і проходимо по resultSet, так як повертатися може декілька об'єктів, за допомогою методу next(). В циклі створюємо об'єкт типу студент та ініціалізуємо всі його поля. Що дістати конкретні дані з resultSet скористаємось методом getString(), як аргумент можна вказати порядковий номер атрибута як ми вказували в запиті, або, що простіше, в казати назву атрибута. Після завершення циклу процедура завантаження даних про студента завершена. Завантаження викладача відбувається аналогічно.

В процесі життєвого циклу інформаційної системи може знадобиться змінити дані про об'єкт то потрібно реалізувати оновлення даних. Для оновлення даних використовується updateStudent(Strudent):

```
private boolean updateStudent(Student student) {
    try {
        String insert = " UPDATE  " + Constant.Student.STUDENT_TABLE +
                " INNER JOIN " + Constant.User.USER_TABLE +
                " ON " + Constant.Student.STUDENT_TABLE + "." + Constant.Student.USERID + " = " +
                Constant.User.USER_TABLE + "." + Constant.User.idUser +
                " SET " + Constant.Student.STUDENT_TABLE + "." + Constant.Student.FIRST_NAME + " = ? " +
","
                + Constant.Student.STUDENT_TABLE + "." + Constant.Student.LAST_NAME + " = ? " + ","
                + Constant.Student.STUDENT_TABLE + "." + Constant.Student.GRADEBOOKNUMBER + " = ? " +
","
                + Constant.Student.STUDENT_TABLE + "." + Constant.Student.GENDER + " = ? " +
                " WHERE BINARY " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =? and " +
Constant.User.USER_TABLE + "." + Constant.User.PASSWORD + " =?;";

        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, student.getFirstName());
        prSt.setString(2, student.getLastName());
        prSt.setString(3, student.getGradebookNumber());
        prSt.setString(4, student.getGender());
        prSt.setString(5, student.getLogin());
        prSt.setString(6, student.getPassword());

        prSt.executeUpdate();
        dbConnection.close();
        return true;
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
        return false;
    }
}
```

Запит на оновлення схожий на запит на збереження. Все починається з ключового слова UPDATE, після вказуємо в якій таблиці будуть відбуватись. Потім після SET перераховуємо атрибути і в кінці вказуємо критерії за якими будуть відбуватися зміни. Далі процес заміни “?” і відправлення запиту до СУБД. Викладач оновлюються аналогічно.

Для видалення студента використовуємо removeStudent(User), де в якості аргументу передається об’єкт User:

```
public boolean removeStudent(User user) {
    String insert = "DELETE FROM " + Constant.User.USER_TABLE + " WHERE " +
Constant.User.USER_TABLE + "." + Constant.User.LOGIN + "= ?" + " and " + Constant.User.USER_TABLE +
"." + Constant.User.PASSWORD + " = ?;";

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, user.getLogin());
        prSt.setString(2, user.getPassword());
```

```
            prSt.executeUpdate();
            dbConnection.close();
            return true;
        } catch (SQLException | ClassNotFoundException e) {
            return false;
        }

    }
```

Для видалення студента достатньо видалити відповідного користувача, видалення інших даних виконає автоматично СУБД. Розглянемо запит на видалення, спочатку вказуємо ключові слова DELETE FROM потім назву таблиці і вкінці після WHERE вказуємо параметри за якими видаляємо. Видалення викладача відбувається аналогічно.

Розглянемо збереження сутності “тест”. Для реалізації цієї функції використовується saveTest(Test):

```
public boolean saveTest(Test test) {
String insert = "INSERT INTO " + Constant.Test.TEST_TABLE + "(" + Constant.Test.NAME + "," +
          Constant.Test.ACCESS + "," + Constant.Test.TIME + "," + Constant.Test.TEACHER_idTEACHER + ")
SELECT ?,?,?, " + Constant.Teacher.idTEACHER + " FROM " + Constant.Teacher.TEACHER_TABLE +
         " INNER JOIN Tester." + Constant.User.USER_TABLE + " ON " +
          Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.USERID + " = " +
          Constant.User.USER_TABLE + "." + Constant.User.idUser +
         " WHERE " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =?;";

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, test.getName());
        if (test.getAccess())
            prSt.setString(2, "1");
        else
            prSt.setString(2, "0");
        prSt.setString(3, String.valueOf(test.getTime()));
        prSt.setString(4, test.getTeacher().getLogin());
        prSt.executeUpdate();
        addQuestion(test);
        addAnswer(test);
        addSubjectHasTest(test);

        dbConnection.close();
        return true;
    } catch (SQLException | ClassNotFoundException e) {
        updateTest(test);
        //e.printStackTrace();
        return false;
    }
}
```

В якості аргументу saveTest приймає об’єкт типу Test. Так як і для попередніх сутностей потрібно створити строкову зміну та ініціалізувати її шаблоном запиту. Форма запиту використовується така ж, як і в попередніх випадках. Після заміни “?” отримуємо запити:

```
INSERT INTO Test(name, access, time, Teacher_idTeacher)
SELECT 'name', 'access', 'time', idTeacher FROM Teacher
INNER JOIN Tester.User ON Teacher.User_idUser = User.idUser
WHERE User.login ='login';
```

Після відправлення запиту до СУБД створюється екземпляр тесту, але поки це тільки назва тесту, для додавання питань та відповідей реалізуються додаткові функції які викликаються після відправлення запиту. Для додавання питань використовується addQuestion(Test), в якості аргументу приймає об’єкт тесту.

```
public boolean addQuestion(Test test) {
    String insert = "INSERT INTO " + Constant.Question.QUESTION_TABLE + "(" +
Constant.Question.QUESTION + "," + Constant.Question.TEST_idTEST + ")" + "SELECT ?," +
Constant.Test.TEST_TABLE + "." + Constant.Test.idTEST + " FROM " + Constant.Test.TEST_TABLE +
            " INNER JOIN Tester." + Constant.Teacher.TEACHER_TABLE +
            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.idTEACHER + " = " +
Constant.Test.TEST_TABLE + "." + Constant.Test.TEACHER_idTEACHER +
            " INNER JOIN Tester." + Constant.User.USER_TABLE +
            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.USERID + " = " +
Constant.User.USER_TABLE + "." + Constant.User.idUser +
            " WHERE " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =? and " +
Constant.Test.TEST_TABLE + "." + Constant.Test.NAME + "=?;";
    for (String question : test.getQuestion()) {

        try {

            PreparedStatement prSt = getDbConnection().prepareStatement(insert);
            prSt.setString(1, question);
            prSt.setString(2, test.getTeacher().getLogin());
            prSt.setString(3, test.getName());

            prSt.executeUpdate();
            dbConnection.close();
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    return true;
}
```

Створення запиту реалізується так само. Але, так як тест може мати декілька питань створюється цикл, який по черзі зберігає питання. Відправляється наступний запит:

```
INSERT INTO Question(question,Test_idTest)
SELECT 'question', Test.idTest
FROM Test INNER JOIN Tester.Teacher ON Teacher.idTeacher = Test.Teacher_idTeacher
INNER JOIN Tester.User ON Teacher.User_idUser = User.idUser
WHERE User.login ='login' and Test.name='name';
```

Для збереження відповідей використовуємо addAnswer(Test):

```
public boolean addAnswer(Test test) {
    if (!test.getAnswer().isEmpty()) {
        for (int i = 0; i < test.getAnswer().size(); i++) {
            String[] answer = test.getAnswer().get(i);
            for (int j = 0; j < answer.length; j++) {
                try {
                    String insert = "INSERT INTO " + Constant.Answer.ANSWER_TABLE + "(" +
Constant.Answer.ANSWER + "," + Constant.Answer.RIGHT_ANSWER + "," +
Constant.Answer.QUESTION_idQUESTION + ")" + "SELECT ?, ?, " + Constant.Question.idQUESTION + "
FROM " + Constant.Test.TEST_TABLE +
                            " INNER JOIN Tester." + Constant.Question.QUESTION_TABLE +
                            " ON " + Constant.Question.QUESTION_TABLE + "." + Constant.Question.TEST_idTEST +
" = " + Constant.Test.TEST_TABLE + "." + Constant.Test.idTEST +
                            " INNER JOIN Tester." + Constant.Teacher.TEACHER_TABLE +
                            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.idTEACHER + " = "
+ Constant.Test.TEST_TABLE + "." + Constant.Test.TEACHER_idTEACHER +
                            " INNER JOIN Tester." + Constant.User.USER_TABLE +
                            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.USERID + " = " +
Constant.User.USER_TABLE + "." + Constant.User.idUser +
                            " WHERE " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =? and " +
Constant.Test.TEST_TABLE + "." + Constant.Test.NAME + "=? and " +
Constant.Question.QUESTION_TABLE + "." + Constant.Question.QUESTION + " =?";
                    PreparedStatement prSt = getDbConnection().prepareStatement(insert);
                    prSt.setString(1, answer[j]);
                    if (test.getRightAnswer().get(i).equals(answer[j])) prSt.setString(2, "1");
                    else prSt.setString(2, "0");
                    prSt.setString(3, test.getTeacher().getLogin());
                    prSt.setString(4, test.getName());
                    prSt.setString(5, test.getQuestion().get(i));
                    prSt.executeUpdate();
                    dbConnection.close();
                } catch (SQLException | ClassNotFoundException e) {
                    System.out.println("Something went wrong");
                }
            }
        }
        return true;
    }
    return false;
}
```

Кожен тест має декілька питань, а кожне питання має декілька відповідей. Тому в об’єкті текст збереження відповідей реалізація за допомогою LinkedList, який зберігає строковий масив в якому зберігається відповіді до одного питання. Тому щоб зберегти відповіді створюємо два цикли один з яких влежується в інший. Перший цикл проходить по LinkedList, а інший по самому масиву. Створення запиту знаходить в тілі вложеного циклу та створення нічим не відрізняється від попередніх. В результаті отримуємо наступний запит:

```
INSERT INTO Answer(answer, right_answer, Question_idQuestion)
SELECT 'answer', '0', idQuestion FROM Test
INNER JOIN Tester.Question ON Question.Test_idTest = Test.idTest I
NNER JOIN Tester.Teacher ON Teacher.idTeacher = Test.Teacher_idTeacher
INNER JOIN Tester.User ON Teacher.User_idUser = User.idUser
WHERE User.login ='Login' and Test.name='Name' and Question.question ='Question'
```

Завантаження тесту виконує loadTest(User) приймає аргумент об’єкт користувача, так як достатньо логіні та паролю для ідентифікації викладача і відповідно його тестів. Створення шаблону та самого запиту відрізняється від попередніх випадків лише назвами таблиць та атрибутів.

```
public ArrayList<Test> loadTest(User user) {

    ResultSet resultSet = null;
    ArrayList<Test> tests = new ArrayList<>();

    Test test = new Test();

    String select = "SELECT " + Constant.Test.TEST_TABLE + "." + Constant.Test.NAME + "," +
Constant.Test.TEST_TABLE + "." + Constant.Test.ACCESS + "," + Constant.Test.TEST_TABLE + "." +
Constant.Test.TIME + " FROM " + Constant.Test.TEST_TABLE +
            " INNER JOIN " + Constant.Teacher.TEACHER_TABLE +
            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.idTEACHER + " = " +
Constant.Test.TEST_TABLE + "." + Constant.Test.TEACHER_idTEACHER +
            " INNER JOIN " + Constant.User.USER_TABLE +
            " ON " + Constant.Teacher.TEACHER_TABLE + "." + Constant.Teacher.USERID + " = " +
Constant.User.USER_TABLE + "." + Constant.User.idUser +
            " WHERE " + Constant.User.USER_TABLE + "." + Constant.User.LOGIN + " =? and " +
Constant.User.USER_TABLE + "." + Constant.User.PASSWORD + " =?;";
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        prSt.setString(1, user.getLogin());
        prSt.setString(2, user.getPassword());

        resultSet = prSt.executeQuery();
        while (resultSet.next()) {
            test.setName(resultSet.getString(Constant.Test.NAME));
            test.setAccess(resultSet.getBoolean(Constant.Test.ACCESS));
```

```
            test.setTime(resultSet.getInt(Constant.Test.TIME));
            test.setSubject(loadSubject(test));
            test.setTeacher(loadTeacher(user));
            test.setQuestion(loadQuestion(test));
            test.setAnswers(loadAnswer(test), loadRightAnswer(test));
             tests.add(test)

        }
        dbConnection.close();
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }

    return tests;
}
```

Так само як і для збереження для питань та відповідей створено додаткові функції loadQuestion(test), та loadAnswer(test). В результаті функції отримуємо наступний запит:

```
SELECT Test.name,Test.access,Test.time FROM Test
INNER JOIN Teacher ON Teacher.idTeacher = Test.Teacher_idTeacher
INNER JOIN User ON Teacher.User_idUser = User.idUser
WHERE User.login ='login' and User.password ='password';
```

Запит для оновлення тесту створюється за допомогою updateTest(Test test):

```
private boolean updateTest(Test test) {
    try {
        String insert = " UPDATE  " + Constant.Test.TEST_TABLE +
                " SET " + Constant.Test.TEST_TABLE + "." + Constant.Test.ACCESS + " = ? " + ","
                + Constant.Test.TEST_TABLE + "." + Constant.Test.TIME + " = ? " +
                " WHERE BINARY " + Constant.Test.TEST_TABLE + "." + Constant.Test.NAME + " =?;";

        PreparedStatement prSt = null;

        prSt = getDbConnection().prepareStatement(insert);
        if (test.getAccess())
            prSt.setString(1, "1");
        else
            prSt.setString(1, "0");
        prSt.setString(2, String.valueOf(test.getTime()));
        prSt.setString(3, test.getName());


        prSt.executeUpdate();
        dbConnection.close();
        updateQuestion(test);
        updateSubjectHasTest(test);

    } catch (SQLException throwables) {
        throwables.printStackTrace();
```

```
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return true;
}
```

Створений запит:

```
UPDATE  Test SET Test.access = 'access' ,Test.time = 'time'
WHERE BINARY Test.name ='name';
```

Для видалення тесту використовуємо функцію removeTest(Test)

```
public boolean removeTest(Test test) {
    String insert = "DELETE FROM " + Constant.Test.TEST_TABLE + " WHERE " +
Constant.Test.TEST_TABLE + "." + Constant.Test.NAME + "= ?;";

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, test.getName());


        prSt.executeUpdate();

        prSt.executeUpdate();
        dbConnection.close();
        return true;
    } catch (SQLException | ClassNotFoundException e) {
        return false;
    }
}
```

Запит для видалення:

```
DELETE FROM Test
WHERE Test.name  = 'name';
```

В цьому розділі ми реалізували інтерфейс між сервером та СУБД. Розглянули, як реалізувати запити на збереження, завантаження, оновлення та видалення даних із бази даних. Ознайомились з синтаксисом мови програмування SQL.

Висновок

В ході дипломної роботи була поставлена задача для розробки автоматизованої системи супроводу навчального процесу. Для вирішення було розглянуто існуючі системи. Було розглянуто архітектури інформаційних систем, та обрана архітектура “сервер додатків”. А також було спроектовано та реалізовано база даних з використанням СУБД MySQL. На даному етапі розробки реалізовано базу даних та сервер для проведення тестів. В майбутньому планується розробка частини клієнта та мобільний додаток, а також реалізація автоматизації контролю присутності студентів на парах.

Література

1. What are the design goals for classroom? URL:https://support.google.com/edu/classroom/forum/AAAAq1rTZJoLJO8SAlhQ1s/?hl=en&msgid=7_Kj06SBBwAJ&gpf=d/msg/google-education/LJO8SAlhQ1s/7_Kj06SBBwAJ (дата звернення 11.05.2020)

2. Елена Тулина Введение в Google Classroom. URL: http://newtonew.com:81/web/vvedenie-v-google-classroom (дата звернення 11.05.2020)

3. Сергей Золотухин 6 систем дистанционного обучения: какую выбрать школе, репетитору, тренеру? URL:https://www.eduneo.ru/3-besplatnye-sistemy-distancionnogo-obucheniya-obzor/ (дата звернення 02.04.2020)

4. About Moodle URL:https://docs.moodle.org/38/en/About_Moodle (дата звернення 02.04.2020)

5. WeChat URL: https://uk.wikipedia.org/wiki/WeChat (дата звернення 12.05.2020)

6. Анастасия Николаенкова Почему нам стоит присмотреться к QR-кодам в 2019 году URL:https://lifehacker.ru/qr-kody-v-2019-godu/ (дата звернення 12.05.2020)

7 Чураков Д. А. “Автоматизированная система обучения и оценки знаний” URL: https://works.doklad.ru/view/5Oyn_El_mAg.html (дата звернення 03.04.2020)

8. Автоматизированные системы обучения (АСО) и их виды URL:https://helpiks.org/5-7119.html (дата звернення 03.04.2020)

9. В.Канаво Методические рекомендации по созданию курса дистанционного обучения через интернет URL:http://www.curator.ru/method.html (дата звернення 03.04.2020)